Mathematical Programming Approaches for Optimal University Timetabling

Bagger, Niels-Christian Fink

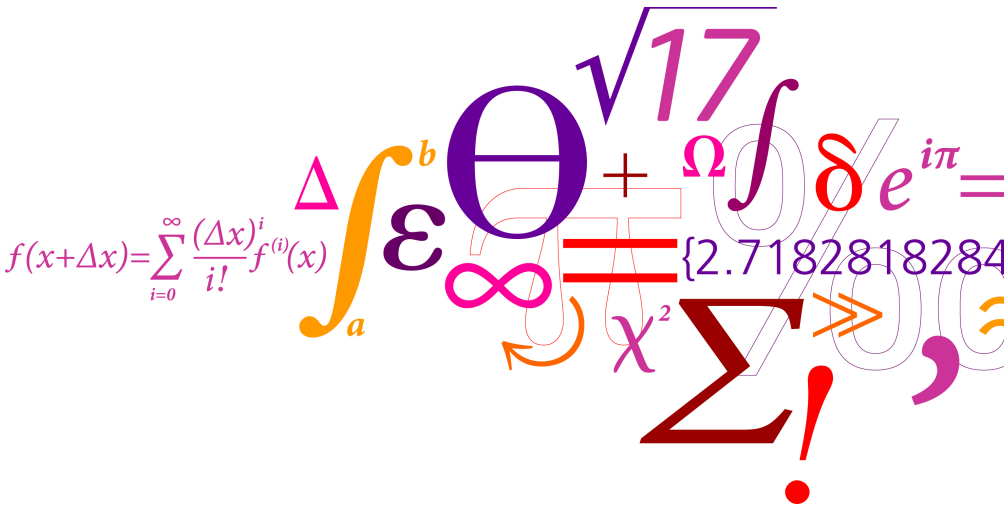
Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Mathematical Programming Approaches for Optimal University Timetabling

PhD Thesis

Niels-Christian F. Bagger
January, 2017



Title:	Mathematical Programming Approaches for Optimal University Timetabling
Type:	PhD Thesis
Date:	January, 2017

Author:	Niels-Christian F. Bagger
Supervisors:	Associate Professor Thomas R. Stidsen Management Science DTU Management Engineering Technical University of Denmark Produktionstorvet, Building 426B DK-2800 Kgs. Lyngby
	Matias Sørensen MaCom A/S Vesterbrogade 48, 1. DK-1620 København V
University:	Technical University of Denmark
Department:	DTU Management Engineering
Division:	Management Science
Address:	Produktionstorvet, Building 426B DK-2800 Kgs. Lyngby
Telephone:	+45 45254800

Abstract

Every semester universities are faced with the challenge of creating timetables for the courses. Creating these timetables is an important task to ensure that students can attend the courses they need for their education. Creating timetables that are feasible can be challenging, and when different preferences are taken into account, the problems become even more challenging. Therefore, automating the processes of generating these timetables is a great help for the planners and the universities. Scheduling and timetabling has been studied before in the literature, and two international conferences are dedicated to this research field.

This thesis considers a University Timetabling problem, more specifically the Curriculum-based Course Timetabling (CTT) problem. The objective of the CTT problem is to assign a set of lectures to time slots and rooms. The literature has focused mainly on heuristic applications which are also apparent in the different surveys. The drawback of the heuristics is that they are problem specific and do not provide any information on the quality of the solutions they generate. The objective of this thesis is to minimize the gap between the best-known upper bounds and the best-known lower bounds for CTT by using Mixed Integer Programming (MIP) based approaches.

We present a total of 15 different MIP based approaches that we have implemented, ranging from Cutting Plane techniques and Lagrangian Relaxation to Benders' Decomposition and Dantzig-Wolfe Decomposition. Most of these implementations did not provide satisfying results. However, they provide valuable insights into the difficulties of the problem. We discuss all the approaches, the difficulties we have encountered, and suggestions on how to bring research further.

Four of the implementations have led to articles submitted to international peer-reviewed journals. The first two articles focus on exact methods and extend each other. The last two focus on generating high-quality lower bounds by applying an extended formulation, which is then decomposed. The articles in this thesis have brought us closer to the goal of closing the gap between the best-known upper and lower bounds for CTT. Though CTT was the problem in focus, the methods implemented here are general enough to be applied for other scheduling problems as well.

Resumé (Danish Abstract)

Hvert semester står universiteter over for udfordringen med at planlægge deres kurser. Denne planlægning er en vigtig opgave for at sikre, at de studerende kan deltage i de kurser der er nødvendige for at komme igennem deres uddannelse. Opgaven med at få skemaerne til at gå op kan være udfordrende i sig selv, og når forskellige præferencer tages i betragtning, bliver opgaven blot endnu vanskeligere. Derfor er automatiserede planlægningssystemer en stor hjælp for planlæggerne og universiteterne. Planlægning og skemalægning er blevet studeret i litteraturen før, og to internationale konferencer er dedikeret til dette område.

Denne afhandling studerer et universitetsskemalægningsproblem, mere specifikt Pensum-baseret Kursus Skemalægningsproblemet (PKS). Formålet med PKS er at tildele et sæt af forelæsninger til ugentlige tidsintervaller og lokaler. I litteraturen er der fokuseret primært på heuristiske metoder. Ulempen ved disse heuristikker er, at de er problem-specifikke og de giver ikke nogen oplysninger om kvaliteten af de løsninger de genererer. Formålet med denne afhandling er at minimere den afstand der er mellem de bedst kendte øvre grænser og de bedst kendte nedre grænser for PKS ved hjælp metoder baseret på matematisk programmering (MP).

Vi præsenterer i alt 15 forskellige metoder baseret på MP, som vi har implementeret. De fleste af disse implementeringer gav ikke tilfredsstillende resultater, men de giver værdifuld indsigt til vanskelighederne ved PKS og ideér til videre forskning. Vi diskuterer alle de metoder, de vanskeligheder vi er stødt på, og forslag til, hvorledes forskningen kan videreføres.

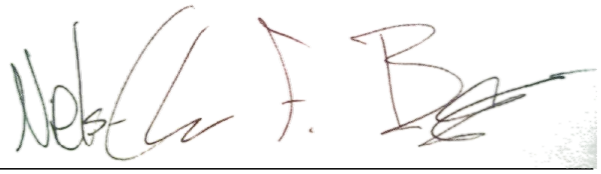
Fire af implementeringerne har ført til artikler indsendt til internationale tidsskrifter. De to første artikler fokuserer på eksakte metoder og ligger i forlængelse af hinanden. De sidste to fokuserer på at generere nedre grænser af høj kvalitet og ligger ogsÃě i forlÃęngelse af hinanden. Artiklerne i denne afhandling har bragt os tættere på målet om at mindske afstanden mellem de bedst kendte øvre og nedre grænser for PKS. Selvom PKS har været problemet i fokus her, så er de metoder der er implementeret generelle nok til at blive anvendt til andre planlægningsproblemer.

Preface

This thesis is part of the requirements for acquiring the degree *Philosophiae Doctor* (Ph.D.) at the Technical University of Denmark. This work has been a collaboration between the Technical University of Denmark and MaCom A/S, a Danish provider of administration software for the educational sector. The project has been financially supported by MaCom A/S and Innovation Fund Denmark (IFD) under the *Industrial Ph.D. Program*. The work has been conducted at the Division Management Science in the Department DTU Management Engineering of the Technical University of Denmark from February 2014 to January 2017. The thesis consists of an introduction and four papers submitted to peer-reviewed journals. As part of the study, a visit to the *Groupe d'Études et de Recherche en Analyse des Décisions* for four and a half month was conducted in the first half of 2015.

The project has been supervised by Associate Professor Thomas R. Stidsen with Matias Sørensen as a co-supervisor.

Kgs. Lyngby, Denmark, January 2017



Niels-Christian F. Bagger

Acknowledgments

First of all, I would like to thank my supervisors, Thomas R. Stidsen and Matias Sørensen for their guidance throughout these last three years. Especially, I would like to thank Thomas for the support, not only on an academic level, but also on a more personal level, and on his endeavor to ensure that I can continue in the academic life after this project.

I would also like to thank Mads Poulsen and Martin Holbøll from MaCom A/S for giving me the opportunity of working on this project. Furthermore, I would like to thank all my colleagues at MaCom for just being fun and friendly. It has been three great years, and I am going to miss the people there.

Big thanks to my colleagues at the Technical University of Denmark (DTU). I look forward to staying there for, hopefully, many years to come. I would like to thank Assistant Professor Evelien van der Hurk and Christina Scheel Persson for their valuable feedback on some of the papers. Thanks to Stefan Røpke for always keeping the door open for me whenever I had any questions. I would also like to thank David Pisinger for guidance, both regarding my career choice and on a more personal level. Lastly, I would like to thank all my colleagues at DTU for welcoming my dog, Freya. She is perhaps the most eager of all to get to work, most likely because she is treated so well by all.

I would also like to thank Professor Guy Desaulniers and Professor Jacques Desrosiers, whom I visited in Montréal for four and a half month during this project. I really appreciate the time and effort they put into me and my project, both during my stay and the time after. Their involvement really exceeded my expectations. I would also like to thank Carole Dufour and Marie Perreault for being really friendly and great to talk to during my stay. I would like to thank the entire GERAD group for their friendliness and also those from the CIRRELT group that I had a chance to meet.

A great thanks go to my family and friends. First, I would like to thank my good friend and colleague Michael Lindahl for joining me on this journey. Michael started his Ph.D. project at the same time as me, and he has been a great support throughout the entire process. I would not have been able to accommodate some of the challenges without him. Thanks to my best friend Stephan Gerdes Høegh for his friendship and emotional support, not just the last three years, but throughout my entire life. Also big thanks to my other best friend Kari Falk Aaen, who has also always been there for me since the day I was born. Great thanks to my mother-in-law for all the help that she has provided, especially during the last part of the writing phase. Last, but definitely not least, I would like to thank my wife and daughter for putting up with me during the last three years, and for joining me during my research stay in Montréal. I have not been easy to live with, especially these last three to six months, and I

would probably have given up along the way without you.

Contents

Part I

Introduction

1 Background

The tasks of generating timetables are frequently occurring at universities. Each semester, events, such as lectures, tutorials, seminars, and exams, need to be scheduled into periods and assigned rooms. The problem is very time consuming to solve manually. Thus there is a need for automated timetabling. Automated Timetabling has long been researched, and there are even two biennial conferences dedicated to this field; the International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT) and the Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA).

To attract more attention to this research area, an International Timetabling Competition was organized in 2002 (ITC2002), where a university timetabling problem was provided. Following the success of ITC2002, a second International Competition was organized in 2007 (ITC2007). Both of the competitions were sponsored by PATAT. The main contribution of ITC2007 was the definitions of University Timetabling problems. McCollum et al. (2010) split the University Timetabling problem into two different problems; the Exam Timetabling problem and the Course Timetabling problem. The Course Timetabling problem is further divided into two subproblems; Post Enrolment-based Course Timetabling (PE-CTT) (Lewis et al., 2007) and Curriculum-based Course Timetabling (CB-CTT) (Di Gaspero et al., 2007). The relation between the problems is illustrated in Figure 1.1.

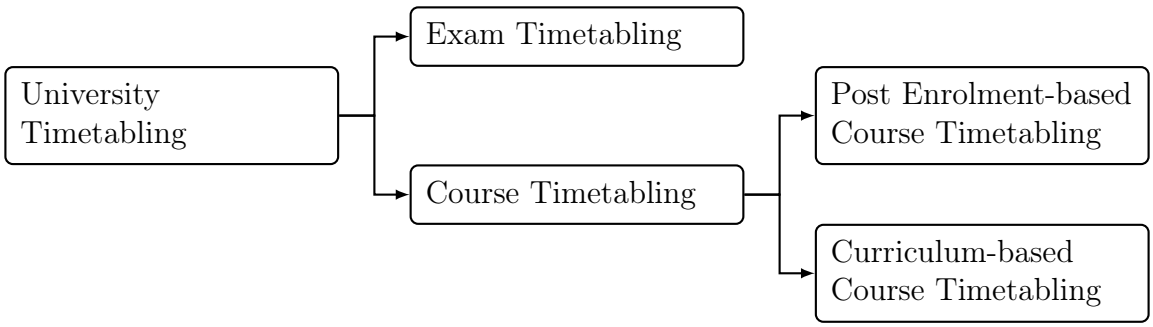


Figure 1.1: The different types of University Timetabling problems presented at ITC2007.

The main difference between PE-CTT and CB-CTT is that each course in PE-CTT consists of a single event, whereas in CB-CTT each course can contain multiple lectures. Another difference is that in PE-CTT it is not allowed to put courses into rooms where the capacity is not large enough. In CB-CTT it is allowed to schedule courses in rooms that are too small at the cost of a penalty in the objective function. The problem that we consider throughout this thesis is the CB-CTT problem defined by Di Gaspero et al. (2007). In the remainder of the thesis, we refer to CB-CTT as CTT. This problem has received most attention in the literature.

One of the reasons for the popularity of this problem is that a website was created as a result of the competition (The Scheduling and Timetabling Research Group at the University of Udine, Italy, 2015). This website has made it possible for researchers to upload instances and compare results.

Most of the research that considers these problems focus on heuristic implementations. The drawback of the heuristics is that they are often problem-specific and do not provide any guarantee of optimality. So provided a solution from a heuristic it is unknown how far from optimality it is unless the optimal solution or a lower bound is known in advance (assuming that it is a minimization problem). If a heuristic is 5% away from optimality, then this may be considered as acceptable, but if the heuristic, for instance, is 60% away from optimality, then maybe the implementation of the heuristic should be reconsidered.

Optimal solutions can often be difficult to obtain o that lower bounds can be used instead. However, the quality of the lower bounds is important. For instance, when this Ph.D. project started in 2014, the gap for one of the data instances from ITC2007 between the best-known solution and the best-known lower bound was more than 65%. This large gap makes the heuristic appear poor in performance, but this is not necessarily the case. During the work for this thesis, we improved the lower bound for that particular instance such that the gap for the same solution is decreased to approximately 15%. Therefore, we focus on methods that either search for the optimal solutions or at least provide lower bounds so the quality of heuristics can be verified.

In the following section 1.1 we describe CTT in details, and in section 1.2 we provide an outline for the thesis. We assume that the reader is familiar with Mixed Integer Programming (MIP) and Operations Research in general.

1.1 Curriculum-based Course Timetabling

In this section, we describe the CTT problem as defined by Di Gaspero et al. (2007) and McCollum et al. (2010) for ITC2007. We are provided with the following; courses, days, time slots, lecturers, rooms, and curricula. Each course is taught by exactly one lecturer, and contains lectures that must all be scheduled in a weekly timetable and assigned rooms. The week is divided into days and each day is divided into time slots which are all equal in size. We refer to a day and time slot pair as a period, so the total number of periods is the number of days multiplied by the number of time slots. The length of one lecture corresponds to one period. A curriculum is a set of courses where for every pair there is a set of students attending both courses. Furthermore, we are given a set of *hard* and *soft* constraints. The weekly schedule and assignment to rooms must fulfill all the hard constraints, which are as follows:

Lectures (L): Every lecture must be scheduled in a period. If two lectures correspond to the same course, then they must be scheduled in different periods. If a lecture is not scheduled, then it is counted as one violation, and if two lectures of the same course are scheduled in the same period, then this is also counted as one violation.

Availability (A): A course can have specific periods defined as unavailable periods. If a lecture from the course is scheduled in an unavailable period, then this is counted as one violation.

Conflicts (C): If two courses are taught by the lecturer of if they belong to the same curriculum, then they cannot have lectures scheduled in the same periods.

Room Occupancy (RO): Every room cannot accommodate more than one lecture in any period. If more than one lecture fulfill in the same room and same period, then this constraint is violated by the number of lectures, minus one.

The problem contains the following four soft constraints, where the goal is to minimize the violations:

Room Capacity (RC): We are allowed to schedule any course into any room. However, it is desired to be able to accommodate as many students as possible when scheduling the courses into rooms. Every room has a capacity, i.e., the number of students that the room can accommodate. If a course is assigned to a room and the number of students attending is larger than the capacity of the room, then the violation is one for each student more than the capacity.

Room Stability (RStab): As the courses contain multiple lectures, it can also be an advantage that the lectures are all scheduled in the same room during the week. For every course, the violation is one for every distinct room that the course is assigned to, minus one.

Minimum Working Days (MWD): For every course, it is preferred to spread the lectures across a predetermined number of days. This number is called *minimum working days*. If the lectures are scheduled in fewer days than the minimum working days, then the violation is one for each day below the minimum working days that the lectures are scheduled.

Isolated Lectures (IL): If two periods belong to the same day and are in consecutive time slots, then we say that the periods are *adjacent*. Consider some curriculum and some course belonging to the curriculum. If the course has a lecture scheduled in a period and no lecture from any of the courses belonging to the curriculum has a lecture scheduled in an adjacent illustrate, then we say that the lecture is *isolated*. For every curriculum, the violation is one for every isolated lecture.

Note that the **IL** constraint is usually referred to as the *curriculum compactness* constraint in the literature. We use the name *isolated lectures* as Bonutti et al. (2012) mentions different ways of defining *curriculum compactness* and they use the name *isolated lectures* for the formulation used here and in ITC2007.

Any feasible timetable must fulfil all the hard constraints, i.e., a timetable is considered feasible if, and only if, all the hard constraints have no violations. The objective is then to find a feasible timetable while minimizing the soft constraints. Each soft constraint has a weight associated such that a single-objective is defined by a weighted sum of all the violated soft constraints.

Burke et al. (2010a) show that fulfilling the hard constraints is $\mathcal{NP}$-complete, which means that the overall problem is $\mathcal{NP}$-hard. Empirical studies also illustrate that the models are hard to solve, even within hours of computational time.

To get an idea of what makes this problem hard to solve, we have tested a MIP model from the paper in chapter 4 without the **IL** constraints. We used Gurobi provided by Gurobi Optimization, Inc. (2016) for these tests. In Table 1.1 we report the results for 21 data instances, which were provided for ITC2007. The first column is the data instance, which is followed by the best-known upper bounds (UB) for CTT. The next three columns (B & B) report the results from solving the model when the constraints **IL** are removed. The first column (Time) reports the running time of the MIP solver to find the optimal solution. The second column (Obj) reports the objective value of the optimal solution for the model, which is a lower bound for CTT. The last column (Gap) is the gap between the objective value and the best-known upper bound for CTT. The last six columns report the statistics of the fractionality of the LP relaxation of the models. We define the fractionality as the number of integer variables that are fractional in the optimal solution to the LP relaxation. In the first three columns (All variables) we report the number for all the variables in the models. Since the Isolated Lectures (**IL**) is specific for the time schedule, we also consider the fractionality of the time schedule. For each course and each period, we add all the variables together that schedules the course in that period. If a course in a specific period is scheduled in one room by 0.3 and some other room in the same period by 0.4, then we sum this together, so the course is scheduled in the period by 0.7 in total, and we consider this as a single variable. In the last three columns (Time variables) we report the fractionality of all these aggregated variables. For both (All variables) and (Time variables) we report the fractionality when **IL** is not included in the model (w/o **IL**) and when **IL** is included in the model (w/ **IL**). In the column (Incr.) we report by how much the inclusion of the **IL** constraints increases the number of fractional variables.

In Table 1.1 we see that the model can be solved within minutes when **IL** is not included. A reason for this can be because of the fractionality of the models since the Branch & Bound algorithm must branch whenever the integer variables are fractional. We see that including **IL** increases the number of all fractional variables more than four times on average, and for the (Time variables) the increase is more than eight times. The gap between the objective values of the model without **IL** and the best-known upper bounds for CTT also illustrate by how much the **IL** constraints impact the objective value.

The problem as it has been presented here is the problem we are considering throughout the entire thesis. However, we briefly present some extensions described byMcCollum et al. (2010) and Bonutti et al. (2012) that could be included to cover a broader variety of universities. These could be either soft or hard and include:

Student Lunch Break: The students should not have a lecture scheduled in at least one time slot around lunch time.

Windows: If two lectures from the same curriculum are scheduled on the same day, and no lectures are scheduled in the time slots between them, then this is referred to as a window. The penalty for these could depend on the lengths of the windows.

Student Min/Max Load: A minimum (or maximum) number of lectures that the students should be scheduled on any day could be specified. If at least one lecture is scheduled on a day and the total number of lectures is below the minimum (above the maximum), then this could be penalized.

Table 1.1: The impact of **IL** for the ITC2007 data set.

					LP Fractionality					
		B & B			All variables			Time variables		
Instance	UB	Time	Obj	Gap	w/o **IL**	w/ **IL**	Incr.	w/o **IL**	w/ **IL**	Incr.
comp01	5	1.8	5	0.0%	192	647	×3.4	57	245	×4.3
comp02	24	24.6	0	100.0%	404	2005	×5.0	64	659	×10.3
comp03	64	13.0	0	100.0%	671	1653	×2.5	158	547	×3.5
comp04	35	14.8	0	100.0%	244	1500	×6.1	12	450	×37.5
comp05	284	1.4	15	94.7%	583	1221	×2.1	189	481	×2.5
comp06	27	170.7	0	100.0%	898	2436	×2.7	201	761	×3.8
comp07	6	312.8	0	100.0%	1038	3165	×3.0	186	965	×5.2
comp08	37	22.3	0	100.0%	229	1809	×7.9	0	549	-
comp09	96	20.6	0	100.0%	298	1536	×5.2	29	483	×16.7
comp10	4	51.7	0	100.0%	699	2713	×3.9	104	843	×8.1
comp11	0	0.4	0	0.0%	148	634	×4.3	48	255	×5.3
comp12	294	1.5	0	100.0%	374	1967	×5.3	75	685	×9.1
comp13	59	15.7	0	100.0%	214	1544	×7.2	0	471	-
comp14	51	19.3	0	100.0%	444	1988	×4.5	74	643	×8.7
comp15	62	13.1	0	100.0%	671	1653	×2.5	158	547	×3.5
comp16	18	64.9	0	100.0%	426	2454	×5.8	48	748	×15.6
comp17	56	48.4	0	100.0%	762	2301	×3.0	155	716	×4.6
comp18	61	0.5	0	100.0%	250	961	×3.8	53	347	×6.5
comp19	57	23.7	0	100.0%	839	1641	×2.0	217	512	×2.4
comp20	4	278.3	0	100.0%	764	2813	×3.7	139	901	×6.5
comp21	74	47.2	0	100.0%	1002	2316	×2.3	232	720	×3.1
Avg.				90.2%			×4.1			×8.3

Travel Distance: If the students need to change building between two lectures that are in adjacent periods, then this could be penalized.

Room Suitability: Some rooms may be unsuitable for some courses, e.g., a lecturer may need chemistry equipment for the lectures, and so the room must contain such equipment. A room could also be defined as unsuitable if the room is too big for the course.

Room Availability: Sometimes rooms are occupied by other activities, e.g., seminars and conferences. Lectures cannot be scheduled in the rooms in the periods where the rooms are unavailable.

Double Lectures: Some courses may require that lectures scheduled on the same day must be in adjacent periods, and also in the same room.

Another extension that McCollum et al. (2010) discuss is the weights of the soft constraints. In CTT used for ITC2007, the weights for each soft constraint is set to a constant value. It could be considered to let the weight depend on the number of students attending the courses or curricula. The *Travel Distance* defined in McCollum et al. (2010) penalizes when students change building. However, some universities may have a large campus, and the walking

distances between the rooms must be taken into account, which is the case at the Technical University of Denmark (Bærentsen, 2012).

1.1.1 Previous Work

In this section, we describe approaches from the literature that has considered CTT. The research on university timetabling problems has, in general, focused on heuristics (Phillips et al., 2015). This focus is also apparent in the overview provided by Bettinelli et al. (2015) and the survey by Pillay (2016).

As MIP solvers are increasing in performance, so is the interest in applying MIP based methods for university timetabling problems (Phillips et al., 2015). As we consider exact and lower bounding methods we provide a brief overview of articles in the literature that consider either exact or lower bounding methods.

Burke et al. (2010a) introduces an exact MIP model of CTT. They formulate the **IL** by using a variable for each curriculum and each period. Burke et al. (2008) remove those variables and instead they have just one variable for each curriculum and each day. The value of this variable is then calculated by adding exponentially many constraints. In Burke et al. (2012) they add a subset of the beforementioned constraints and then add the remaining dynamically whenever they are violated. Burke et al. (2010b) takes the model from Burke et al. (2010a) and split it into two stages. The first stage is to schedule the courses into the periods, i.e., ignore the room assignments. However, it is ensured that the time schedule is feasible by not scheduling more lectures in any period than the number of rooms available. In the second stage, they take the period schedule and fix the model either completely or partially to the selected periods and then solve the full model. This approach is executed iteratively.

Splitting the problem into two stages is also considered by Lach and Lübbecke (2008) and Lach and Lübbecke (2012), where the problem is also split into two stages; the first stage creates the time schedule and the second stage makes the room assignment given the time schedule. Lach and Lübbecke (2012) also show how the **RC** constraints can be added to the first stage problem by grouping the rooms together according to their capacities. Then the first stage problem schedules the courses into periods and capacities.

Hao and Benlic (2011) considers the first stage problem of Lach and Lübbecke (2012). They make a decomposition by relaxing some of the constraints such that the problem can be divided into subproblems. They then compute a lower bound for each subproblem and sum them up to get a lower bound for the overall problem.

Cacchiani et al. (2013) also compute lower bounds. They do this by splitting the problem into two parts where one part considers the constraints **MWD** and **IL** and the other part considers the constraints **RC** and **RStab**. A lower bound is then calculated by summing up lower bounds for the two parts. The part which considers the constraints **MWD** and **IL** can be time-consuming to solve. So they apply a Dantzig-Wolfe decomposition of their model such that the pricing problem is decomposable by days and solve the model by column generation.

Asín Aschá and Nieuwenhuis (2014) proposes multiple satisfiability encodings. They start off by treating the soft constraints as hard constraints and solve the problem as a pure satisfiability problem. Then they *relax* the constraints one by one and move towards a weighted partial maximum satisfiability encoding.

In the paper in chapter 4 we provide more details on some of the methods from the literature. For an even more comprehensive overview of the literature regarding CTT, we refer to Bettinelli et al. (2015).

1.2 Thesis Outline

This thesis is divided into three main parts; Part I Introduction, Part II Exact Methods and Part III Lower Bounding Methods. Part I is the introduction to the thesis and covers the description of the problem, the scientific contributions and the conclusions of the work. In chapter 1 we provide the description of the problem that has been considered throughout the work for this thesis, and related work that has been applied. In chapter 2 we summarize the scientific contributions and the conclusion of the work for this thesis. Furthermore, we provide suggestions for future research. The last chapter 3 of part I is a summary of all 15 different approaches that we have implemented and tested during the work for this thesis. The chapter is not necessary to read but is included for the readers that are interested in the details of all the approaches that we have implemented. Part II and part III constitute the majority of the thesis, and both consist of two articles. The articles in part II focuses on exact methods, and the articles in part III focuses on lower bounding methods.

2 Scientific Contributions

The scientific contributions are here summed up for the four papers. All four papers are submitted to international peer-reviewed journals. The first two papers focus on exact methods. A common approach to solving the problem is to divide it into two parts; a *Time Scheduling* problem and a *Room Allocation* problem. Then the Time Scheduling problem is solved, and the solution is provided for the Room Allocation problem to generate a complete solution. This approach can be iterated. The drawback of this approach is that we lose the guarantee of optimality. So the focus in this thesis has been on exact methods in the first two papers and then lower bounding methods in the last two papers. In the following we describe the four papers. Section 2.1 contains our conclusions of the work conducted for this thesis and in section 2.2 we provide suggestions for future research.

Chapter 4: Flow Formulations for Curriculum-based Course Timetabling This paper combines the two components, the *Time Scheduling* problem and the *Room Allocation* problem, into two exact formulations, which are solved by a generic MIP solver. The first formulation is based on an underlying *minimum cost flow* (MIN) problem. The second formulation is based on a *multi-commodity flow* (MULT) problem. The MIN problem is known to contain the integrality property, and hence being solvable in polynomial time, but the MULT problem is $\mathcal{NP}$-hard in general. However, we proved that it suffices to include the LP-relaxation of MULT in the model. For both of these formulations, the result is that the number of integer variables is significantly lower than other exact formulations in the literature at the cost of many continuous variables.

Compared to other approaches in the literature that provide both lower and upper bounds the MIN formulation provides the best performance on the data instances from ITC2007. We also compared the flow formulations with the basic MIP model which we present in section 3 on a total of 32 instances. The results showed that the reformulations outperform the basic model both on the lower and upper bounds. Here the MIN formulation obtained a lower bound which is at least as good as MULT and the basic model on 28 of the instances, and for 11 of these instances, MIN obtained a strictly better bound than the other two. On 24 of the instances the MIN formulation obtained an upper which is at least as good as the other two, and for 12 of these the upper bound is strictly less than for the other two formulations. Out of the 32 instances, six of them are still open. The MIN formulation improved the lower bound of one of them from 101 to 142. We believe that other approaches from the literature based on the basic model can benefit from these reformulations.

The MULT formulation was submitted as an extended abstract to the peer-reviewed MISTA conference in 2015 (Bagger et al., 2015). The full paper with both methods is submitted to *Annals of Operations Research* and contributes with:

- Two new formulations that outperform the basic formulation.
- An improvement of the lower bound for one out of the six instances that are still open by more than 40%.

Chapter 5: Benders' Decomposition for Curriculum-based Course Timetabling In the previous paper *Flow Formulations for Curriculum-based Course Timetabling*, two formulations were provided for CTT such that a large part of the variables could be relaxed to continuous variables. The paper *Benders' Decomposition for Curriculum-based Course Timetabling* expands on one of the formulations by projecting out all these continuous variables. Then a Benders' Decomposition algorithm is implemented, which is the first time to our knowledge that a full Benders' Decomposition algorithm is implemented for CTT, i.e., where Benders' cuts are generated dynamically as they are violated. We also implemented a heuristic to generate upper bounds based on solving a series of *Minimum Cost Maximum Flow* problems as the solutions produced inside the decomposition were usually infeasible. The main focus of the heuristic was to gain feasibility, so we believe that there is a potential here to improve the implementation further.

We compared the decomposition with other approaches on a total of 38 real-life instances. Out of these 38 instances, 12 of them are still open, and our implementation improved the lower bound on eight of these instances. Six of the open instances are significantly larger and more difficult to solve than the other 32. For these six instances our decomposition is the first MIP-based approach that has been applied, and the first time lower bounds have been calculated.

We compared Benders' Decomposition on the large instances with MULT since no other MIP-based methods have been applied. These tests illustrated that the benefits of Benders' Decomposition are more apparent for large data instances. Solving the root node LP with MULT had a running time of more than half an hour for three of the instances, and for the three other instances, the running time was more than one and a half hour. For our decomposition, the longest running time is less than four minutes. On average the speed-up was more than 30 times, and the improvements of the lower bounds were 14%. For the upper bounds, MULT were only able to obtain a feasible solution for four of the instances, which Benders' Decomposition improved by 35% on average. Furthermore, the decomposition was able to obtain solutions for all instances.

The paper is submitted to *Computers & Operations Research* and contributes with:

- The first Benders' Decomposition algorithm for CTT.
- First time that the lower bounds are calculated for six large instances.
- Improvement of the lower bounds for eight out of 12 of the real-life instances that are still open.

Chapter 6: Daily Course Pattern Formulation and Valid Inequalities for the Curriculum-based Course Timetabling Problem The previous two papers focused on the improvement of exact methods and provided methods to combine the Time Scheduling problem and the Room Allocation problem. Empirical studies have shown that the Time

Scheduling problem is the most time-consuming problem of the two components. So in this paper, we focus on the Time Scheduling problem as any improvements on this component can be applied to the two previous approaches. In the paper, *Daily Course Pattern Formulation and Valid Inequalities for the Curriculum-based Course Timetabling Problem* a pattern formulation is provided. For each course and each day we enumerate all the patterns that are possible for the course to be assigned on that day. In our model, there is a binary variable for each of these patterns for each course and each day.

The benefit of the pattern formulation is that we can preprocess the model to remove variables. We implement multiple preprocessing techniques where one is based on solving auxiliary *maximum flow* problems. We also generate valid inequalities, which are difficult to derive for the basic formulation. Some of these inequalities come from generating a conflict graph for the variables, and we show how this graph can be constructed by extending the preprocessing techniques. We discuss in the paper that one of the benefits of the pattern formulation is that it is more flexible regarding adding additional constraints or penalties than the basic model.

We compared the formulation to other lower bounding approaches from the literature on 21 real-life instances from ITC2007, and show that the pattern formulation has a better performance. Four of the instances are still open, and our formulation improves the lower bound of three of them. The paper is submitted to *Journal of Scheduling* and contributes with:

- A new lower bounding formulation of CTT that outperforms other approaches from the literature.
- Implementation of novel preprocessing and clique graph generation techniques.
- Improvements of the best-known lower bound for three out of the four real-life instances that are still open.

Chapter 7: Dantzig-Wolfe Decomposition of the Daily Pattern Formulation for Curriculum-based Course Timetabling Dantzig-Wolfe Decomposition has been applied to CTT before. However, they are all based on the basic formulation. As a stronger formulation is provided in the previous paper *Daily Course Pattern Formulation and Valid Inequalities for the Curriculum-based Course Timetabling Problem*, then we use this formulation for the decomposition. We apply the decomposition such that there is a pricing problem for each day and we solve the LP-relaxation of the master problem by Column Generation. The decomposition puts the formulation of the isolated lectures into the pricing problems, which is an advantage as it was shown in section 1.1 that these soft constraints are the ones that are most difficult.

We provide a preprocessing technique that can be applied in an iteration of the Column Generation algorithm and also show how the technique can be extended to generate inequalities. The empirical study shows that our preprocessing implementation can remove almost half of the variables from the model on average. Applying this technique to other scheduling problems could be interesting.

We implement a Local Branching algorithm to solve the pricing problems by using previously generated columns. To the best of our knowledge, this is the first time Local Branching is implemented in a pricing problem in Column Generation, though the nature of the Column Generation algorithm fits perfectly with Local Branching. As Local Branching is a general

framework and easy to implement it can be considered for other problems solved by Column Generation as well. We show that more than 90% of the time of the Column Generation algorithm is spent in the pricing problems. So we suggest any future research, to focus on improving the running time of the pricing problems.

We compared the decomposition to other approaches from the literature. The lower bounds obtained are higher than for most other approaches except for the pattern formulation in the previous paper. However, for the four instances from ITC2007 that are still open we obtain a higher lower bound for all of them, which decreases the average gap to the best-known upper bounds from 24% to 11%. The paper is submitted to *European Journal of Operational Research* with the following contributions:

- A new Dantzig-Wolfe Decomposition and Column Generation algorithm for CTT.
- Novel preprocessing and inequality generation for the pricing problem.
- The first time Local Branching is applied in a pricing problem inside a Column Generation algorithm.
- Improvements of the best-known lower bounds for all four real-life instances from ITC2007 that are still open.

2.1 Conclusion

Due to the second international timetabling competition in 2007 (ITC2007), the Curriculum-based Course Timetabling (CTT) problem has received a lot of attention. The CTT problem consists of assigning courses into periods and rooms. For University Timetabling problems, in general, most literature has focused on heuristic applications which are also apparent in the different surveys. The heuristics are attractive in real-world settings as they are usually fast. The drawback of the heuristics is that they are problem-specific and do not provide information on how far they are from optimality. For the competition 21 data instances were provided where four of them are still *open*, meaning that for these four instances, the best-known lower bounds do not equal the best-known upper bounds. The objective of this thesis has been to minimize the gap between the best-known upper bounds and the best-known lower bounds for CTT by using Mixed Integer Programming (MIP). A total of 15 different MIP based formulation and methodologies have been implemented and tested during this work. Four of these implementations led to article submissions for peer-reviewed international journals.

Most of the MIP-based approaches in the literature split the problem into two components; a Time Scheduling problem and a Room Allocation problem. The Time Scheduling problem consists of scheduling the course into periods, and the Room Allocation problem assigns courses to rooms. The Time Scheduling problem is commonly solved first, and the solution is then provided to the Room Allocation problem. The first article we submitted focused on combining the two components into one model. Two formulations were provided that improved the performance of a generic MIP solver, both regarding the lower and upper bounds. The second article expanded on the results from the first article by applying a Benders' Decomposition on one of the provided formulations. The results showed improvements on the lower bounds compared to literature. The method was also tested on six large data instances where MIP based approaches

have not been applied before. On these instances, the decomposition improved the performance of the MIP solver significantly on both the lower and upper bounds.

The last two articles focused on improving the lower bounds by considering the Time Scheduling problem as empirical studies in literature have shown that this is the most time-consuming part. In the first of the two articles, a pattern formulation is implemented by enumerating all possible patterns for each course and each day. The pattern formulation provided stronger lower bounds compared to the literature. In the last article, we expanded on the pattern formulation by applying a Dantzig-Wolfe Decomposition of the model and solved it by Column Generation. The benefit of the decomposition was that the formulation of some of the hardest soft constraints was put into the pricing problems, which are smaller and easier to solve. The decomposition further improved the bounds for the four instances from ITC2007 that are still open.

The articles in this thesis have brought us closer to the goal of closing the gap between the best-known upper and lower bounds for CTT. Though CTT was the problem in focus, the methods implemented here are general enough to be applied for other scheduling problems.

2.2 Future Research

We have implemented and tested different MIP based approaches on CTT, leading to four submitted articles. In chapter 3 we describe additional implementations that we have tested, which were not all successful for CTT. In total, we report 15 different formulations, methods and implementations. This vast amount of implementations shows how difficult this problem is, and that further research is needed. The theory and notes on the implementations are provided, and it could be interesting to see if other scheduling problems can benefit from these approaches, or which changes to the methods that are needed for them to be successful for CTT.

Other suggestions for future research is to improve the approaches from our submitted articles. In the second article, we consider a Benders' Decomposition. We implemented a heuristic to obtain feasible solutions but saw that our implementation did not improve the upper bounds. The focus of the heuristic was to obtain a feasible solution by assigning rooms provided that the time schedule was feasible. Therefore, we suggest considering implementing a heuristic which also considers improving the solutions, for instance by also making changes to the time schedule. In the last article, we applied a Dantzig-Wolfe Decomposition. For some of the instances that we tested the implementation of the Column Generation algorithm was too time-consuming to be embedded in a full Branch & Price algorithm. As more than 90% of the running time was spent in the pricing problems we suggest that solution methods for these problems are researched further.

Some methods that we have briefly tested for the pricing problems include Dynamic Programming, Constraint Programming, Lagrangian Relaxation and Benders' Decomposition. However, we have not studied these implementations enough for us to draw conclusions, which is why we have not included them in the description of the implemented approaches. Another interesting study could be to consider why some of the instances are significantly more time consuming than others, for instance by examining the feature space suggested by Smith-Miles et al. (2014). This information could be useful in the development of solution approaches.

When the pricing problems can be solved in a reasonable amount of time, we would like to see the Benders' Decomposition algorithm included in the Dantzig-Wolfe Decomposition. One way to include Benders' Decomposition could be to solve the room allocation problem in another pricing problem and then add the Benders' feasibility cuts in the master problem to connect the pricing problems. Another possibility is to use the current implementation as the lower bounding problem in a Branch & Price algorithm and apply branching rules on the time schedule. Then switch to the Benders' Decomposition algorithm for nodes that are deep enough in the search tree.

3 Implemented Approaches

In this chapter we provide an overview of all the methods that has been tested during this work for the CTT problem. Not all methods have been equally successful. However, the knowledge is useful for the timetabling community to get an overview of methods that are either successful, should be avoided or needs further research. Furthermore, even though some of the approaches have not been successful for CTT, it could be interesting to see if other scheduling or timetabling problems can benefit from these methods. Before describing all the approaches that have been tested, we first provide a basic (MIP) model of the problem. All the tested methods refer to this model as the basis.

The set of courses is denoted $\mathcal{C}$. For each course $c \in \mathcal{C}$, the number of lectures to be scheduled is denoted as L_c. For the periods we have the set of days, $\mathcal{D}$, and the set of time slots, $\mathcal{T}$. For each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ we let the parameter $F_{c,d,t}$ take value one if the course is available in the specific period, and zero otherwise. The set of rooms is denoted $\mathcal{R}$. We let $x_{c,d,t,r}$ be a binary variable taking value one if course $c \in \mathcal{C}$ is scheduled on day $d \in \mathcal{D}$ in time slot $t \in \mathcal{T}$ in room $r \in \mathcal{R}$, and zero otherwise. To ensure that the constraint **L** is not violated we sum over all the binary variables associated with one course and add a constraint that the sum must equal L_c:

$$\sum_{d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} x_{c,d,t,r} = L_c, \quad \forall c \in \mathcal{C} \tag{3.1}$$

This constraint only ensures that all lectures are scheduled, but not that they are scheduled in different time slots. We ensure this by the following constraints, where we also include the **A** constraint:

$$\sum_{r \in \mathcal{R}} x_{c,d,t,r} \leq F_{c,d,t}, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.2}$$

To fulfil the constraint **C** we first construct a graph where every node in the graph corresponds to a course. If two courses are taught by the same lecturer or belong to the same curriculum, then the corresponding nodes are connected by an edge. An example of the graph is illustrated in Figure 3.1.

We then enumerate a set of *course cliques* Γ where a course clique $\gamma \in \Gamma$ is a set of courses $\mathcal{C}_\gamma \subseteq \mathcal{C}$ such that each pair of courses in $\mathcal{C}_\gamma$ is conflicting. To generate the cliques we use the algorithm described by Bron and Kerbosch (1973) that enumerates all maximal cliques in a graph. For every edge in the graph the two courses corresponding to the nodes of the edge are both contained in at least one clique together. The for each clique we add the constraint that at most one of the courses in the clique can be scheduled in any period:

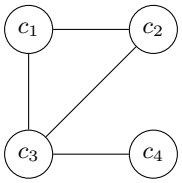


Figure 3.1: Conflict graph where a curriculum consists of courses c_1, c_2 and c_3 while c_3 and c_4 are taught by the same lecturer. The figure is taken from the paper in chapter 6

$$\sum_{c\in\mathcal{C}_\gamma, r\in\mathcal{R}} x_{c,d,t,r} \leq 1, \quad \forall\gamma\in\Gamma, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.3}$$

We define the set $\Gamma_c \subseteq \Gamma$ as the set of cliques that contain course $c \in \mathcal{C}$. The last of the hard constraints to fulfill is the **RO** constraint:

$$\sum_{c\in\mathcal{C}} x_{c,d,t,r} \leq 1, \quad \forall r\in\mathcal{R}, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.4}$$

Any integer solution of the x variables fulfilling constraints (3.1) – (3.4) corresponds to a feasible timetable. Next, we formulate the soft constraints. We let S_c be the number of students attending course $c \in \mathcal{C}$ and C_r be the capacity of room $r \in \mathcal{R}$. Then the violation of the **RC** constraint can be calculated as:

$$\sum_{c\in\mathcal{C}, d\in\mathcal{D}, t\in\mathcal{T}, r\in\mathcal{R}} (S_c - C_r)^+ x_{c,d,t,r} \tag{3.5}$$

where $(x)^+$ for any real number x is defined as $(x)^+ := \max\{0, x\}$. For the **RStab** constraint we introduce a binary variable $z_{c,r}$ for each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ which takes value one if c is assigned to r at least once:

$$\sum_{d\in\mathcal{D}, t\in\mathcal{T}} x_{c,d,t,r} \leq L_c z_{c,r}, \quad \forall c\in\mathcal{C}, r\in\mathcal{R} \tag{3.6}$$

Then we can calculate the violation of the **RStab** constraint as follows:

$$\sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}} z_{c,r} - 1 \right) \tag{3.7}$$

For the **MWD** constraint we let the binary variable $t_{c,d}$ take value one if course $c \in \mathcal{C}$ has at least one lecture scheduled on day $d \in \mathcal{D}$, and zero otherwise:

$$t_{c,d} \leq \sum_{t\in\mathcal{T}, r\in\mathcal{R}} x_{c,d,t,r}, \quad \forall c\in\mathcal{C}, d\in\mathcal{D} \tag{3.8}$$

For each course $c \in \mathcal{C}$, we let the variable w_c calculate the violation of the **MWD** constraint:

$$w_c + \sum_{d\in\mathcal{D}} t_{c,d} \geq D_c^{\min}, \quad \forall c\in\mathcal{C} \tag{3.9}$$

The set of curricula is denoted $\mathcal{Q}$, and for each curriculum $q \in \mathcal{Q}$ the set of courses belonging to q is denoted $\mathcal{C}_q \subseteq \mathcal{C}$. Similarly, we let the set of curricula that course $c \in \mathcal{C}$ belongs to be denoted as $\mathcal{Q}_c$. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ we let the binary variable $s_{q,d,t}$ take value one if q has an isolated lecture at day d in time slot t, and zero otherwise. For $t \in \mathcal{T}$ we denote the time slot that is right before as $t-1$ and we denote the time slot right after t as $t+1$. For ease of notation, we define $x_{q,d,t}$ for each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ as follows:

$$x_{q,d,t} := \sum_{c \in \mathcal{C}, r \in \mathcal{R}} x_{c,d,t,r}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.10}$$

If t is the first time slot then we define $x_{q,d,t-1}$ to be zero and if t is the last time slot then we define $x_{q,d,t+1}$ to be zero. Then we can calculate the isolated lectures as follows:

$$s_{q,d,t} \geq x_{q,d,t} - x_{q,d,t-1} - x_{q,d,t+1}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.11}$$

Let $W^{\mathbf{RC}}$, $W^{\mathbf{RStab}}$, $W^{\mathbf{MWD}}$ and $W^{\mathbf{IL}}$ be the non-negative weights for the constraints **RC**, **RStab**, **MWD** and **IL** respectively. Then, we express the objective function as a weighted sum of the soft constraints to be minimized:

$$\begin{aligned} & W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} (S_x - C_r)^+ x_{c,d,t,r} \\ & + W^{\mathbf{RStab}} \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) \\ & + W^{\mathbf{MWD}} \sum_{c \in \mathcal{C}} w_c \\ & + W^{\mathbf{IL}} \sum_{q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}} s_{q,d,t} \end{aligned} \tag{3.12}$$

We have tested a total of 15 different method for CTT during this thesis. Figure 3.2 illustrates all the 15 methods which have been divided into five frameworks; Mixed Integer Programming, Lagrangian Relaxation, Benders' Decomposition, Cutting Planes and Dantzig-Wolfe Decomposition.

In section 3.1 we provide an overview of the framework *Mixed Integer Programming*. Here we tested four MIP models as alternatives to the basic MIP model which has resulted in two papers. In section 3.2 the *Lagrangian Relaxation* framework is described. We tested three different approaches none of which resulted in papers. The framework *Benders' Decomposition* is described in section 3.3. Three methods were tested within this framework which resulted in one paper. In section 3.4 and overview of three methods tested within the framework *Cutting Planes* is provided where none resulted in papers. The last framework *Dantzig-Wolfe Decomposition* is described in section 3.5. Three different methods were tested within this framework which resulted in one paper.

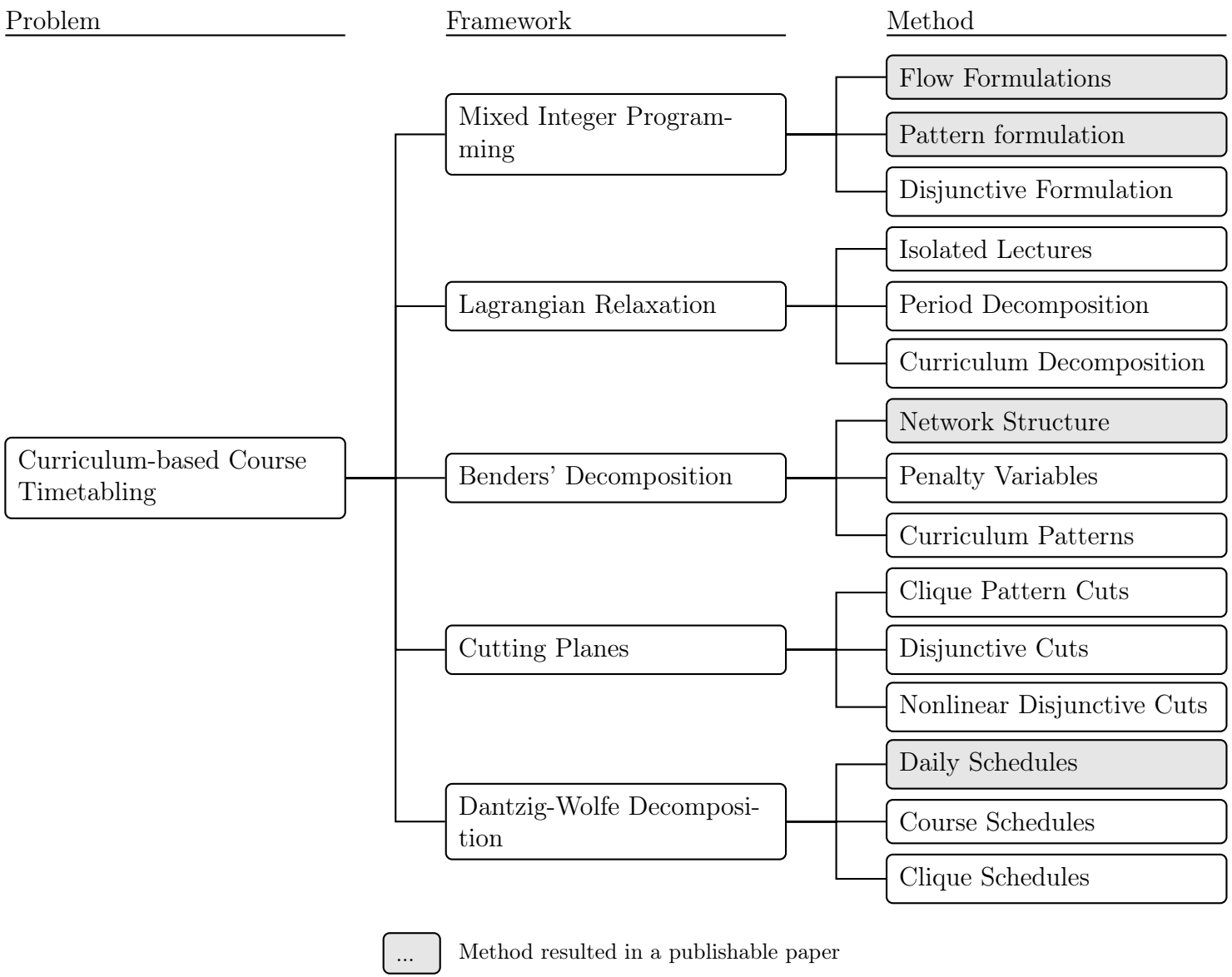


Figure 3.2: Overview of the tested approaches. The different approaches has been divided into five frameworks; Mixed Integer Programming, Lagrangian Relaxation, Benders' Decomposition, Cutting Planes and Dantzig-Wolfe Decomposition

3.1 Mixed Integer Programming

In this section we describe some alternatives to the basic model. We have tested four alternative formulations to the basic model; two formulations based on underlying flow networks, a pattern based formulation where each pattern corresponds to an entire time schedule for one course on one day and the last formulation is based on reformulating the modelling of the isolated lectures. An overview of the flow formulations and the pattern formulation are provided in section 2. A detailed description for the flow formulations is provided in the paper in chapter 4 and the details of the pattern formulation is provided in the paper in chapter 6. In the following section 3.1.1 we describe how the isolated lectures can be reformulated in a disjunctive model.

3.1.1 Disjunctive Formulation

The idea of the disjunctive formulation is to reconsider the formulation of the isolated lectures (3.11). We examine the solution space of the Linear Programming (LP) relaxation of the problem. Consider the constraint (3.11) for a curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ as well as the non-negativity constraint $s_{q,d,t} \geq 0$. Consider a (x, y, z)-coordinate system. The x coordinate corresponds to the value of the $x_{q,d,t}$ variable. The y coordinate is equal to $x_{q,d,t-1} + x_{q,d,t+1}$ and the z coordinate corresponds to the value of the $s_{q,d,t}$ variable. In Figure 3.3 the constraint (3.11) together with the non-negativity constraint are illustrated in the unit cube where the hyperplanes of the constraints are marked in grey.

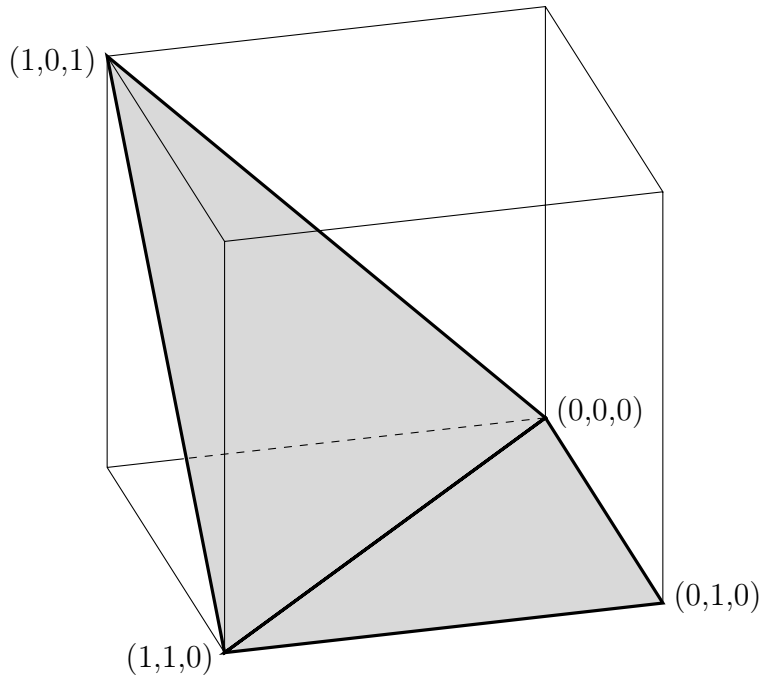


Figure 3.3: Illustration of the isolated lecture constraints in the unit cube

Note that four points are emphasized in Figure 3.3; the origin $(0, 0, 0)$ and $(0, 1, 0)$, $(1, 1, 0)$ and $(1, 0, 1)$. These are the important points as they represent the integer solutions, and in the first three points there is no isolated lecture, and in the last point, there is an isolated lecture.

As the z coordinate corresponds to the $s_{q,d,t}$ variable which is minimised then the value of the $s_{q,d,t}$ variable in the optimal solution to the LP relaxation will be on one of the hyperplanes. This means that for any solution where the (x, y) coordinates are a convex combination of $(0, 0)$, $(0, 1)$ and $(1, 1)$ the z coordinate will be set to zero. It is only necessary for the z coordinate to be zero in the integer (x, y)-points $(0, 0)$, $(0, 1)$ and $(1, 1)$, but not in any fractional point which is a convex combination of the three points. In Figure 3.4 an alternative formulation is illustrated where the z coordinate can only be zero in the unit cube in the integer points (x, y)-points $(0, 0)$, $(0, 1)$ and $(1, 1)$ and in the convex combination of $(0, 0)$ and $(0, 1)$ and the convex combination of $(0, 1)$ and $(1, 1)$. Everywhere else inside the unit cube the z coordinate must be strictly positive.

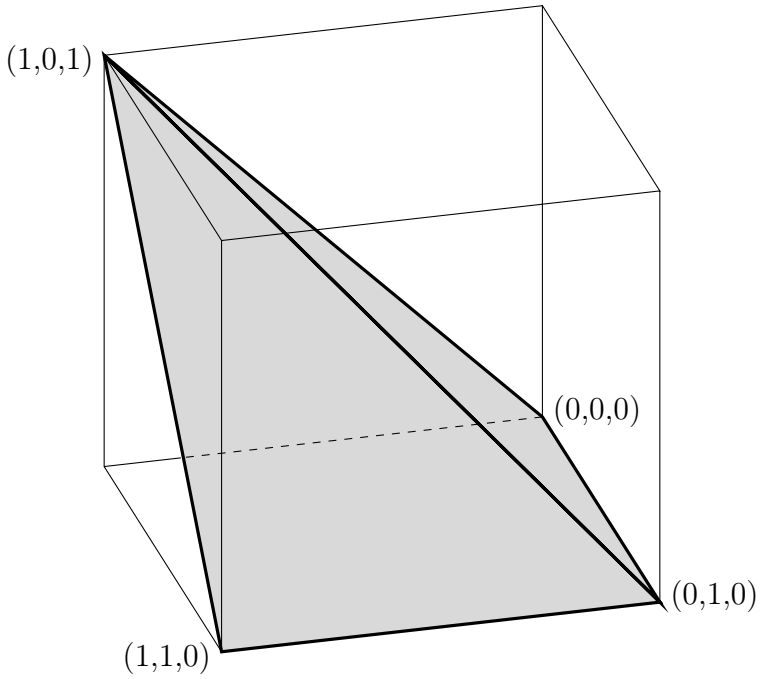


Figure 3.4: Illustration of the disjunctive formulation of the isolated lectures in the unit cube

In Figure 3.4 the solution space of the LP relaxation in the unit cube is the convex combination of the points $(0, 0, 0)$, $(1, 0, 1)$ and $(0, 1, 0)$ or the convex combination of the points $(0, 1, 0)$, $(1, 0, 1)$ and $(1, 1, 0)$. The obstacle with the formulation in Figure 3.4 is that it is non-convex, so we cannot make an LP formulation of this. The line which is a convex combination of $(0, 1, 0)$ and $(1, 0, 1)$ is the intersection of the two convex combinations. The line projected down to the (x, y)-space is the hyperplane $x + y = 1$. So if we consider some solution, we need to know on which side of this hyperplane in the (x, y)-space the solution is. We introduce a binary variable $y_{q,d,t}$ which takes value one if the solution is on the $x + y < 1$ side of the hyperplane and zero if it is on the $x + y > 1$ side. As we cannot formulate $<$ or $>$ constraint in LP models we calculate the value of $y_{q,d,t}$ as follows:

$$x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} \leq 3 - 2y_{q,d,t} \tag{3.13}$$
$$x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} \geq 1 - y_{q,d,t} \tag{3.14}$$

If the point is on the $x + y < 1$ side of the hyperplane in the (x, y)-space then we use the hyperplane spanned by the three points $(0, 0, 0)$, $(1, 0, 1)$ and $(0, 1, 0)$; $x - z = 0$. If the point is on the $x + y > 1$ side of the hyperplane in the (x, y)-space then we use the hyperplane spanned by the three points $(0, 1, 0)$, $(1, 0, 1)$ and $(1, 1, 0)$; $y + z = 1$. In an LP model this can be formulation as follows:

$$s_{q,d,t} \geq x_{q,d,t} + y_{q,d,t} - 1 \tag{3.15}$$
$$s_{q,d,t} \geq 1 - y_{q,d,t} - x_{q,d,t-1} - x_{q,d,t+1} \tag{3.16}$$
$$s_{q,d,t} \geq 0 \tag{3.17}$$

If $y_{q,d,t} = 1$ then constraint (3.15) is *activated* and (3.16) becomes *inactive* and opposite for $y_{q,d,t} = 0$. Note that for any point that is on the hyperplane $x + y = 1$ then $y_{q,d,t}$ can be either zero or one. However, as we know that the variables $x_{q,d,t}$, $x_{q,d,t-1}$ and $x_{q,d,t+1}$ then we can formulate the following disjunction of the model:

$$\left\{ \begin{array}{c} x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} \leq 1 \\ s_{q,d,t} \geq x_{q,d,t} \end{array} \right\} \quad \bigvee \quad \left\{ x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} \geq 2 \right\} \tag{3.18}$$

The disjunction (3.18) corresponds to replacing the right-hand side of (3.14) by $2 - 2y_{q,d,t}$. Introducing this disjunction makes constraint (3.16) redundant as there can only be an isolated lecture in the left branch, i.e., when $y_{q,d,t} = 1$. The disjunction (3.18) also means that we implicitly minimize the value of $y_{q,d,t}$ which makes the constraint (3.13) redundant, thus the disjunctive formulation is as follows:

$$x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} + 2y_{q,d,t} \geq 2 \tag{3.19}$$
$$x_{q,d,t} + y_{q,d,t} - s_{q,d,t} \leq 1 \tag{3.20}$$
$$s_{q,d,t} \geq 0 \tag{3.21}$$

The downside about the formulation (3.19) – (3.21) is that it requires $O(|\mathcal{Q}||\mathcal{D}||\mathcal{T}|)$ extra binary variables. Another downside is that the LP relaxation is weaker compared to the LP relaxtion of the basic model. If $x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1}$ is at least two then both of the formulations does not provide a higher lower bound for $s_{q,d,t}$ than zero. We assume that the sum is less than two and isolate $y_{q,d,t}$ in (3.19):

$$y_{q,d,t} \geq 1 - \frac{1}{2}\left(x_{q,d,t} - x_{q,d,t-1} - x_{q,d,t+1}\right) \tag{3.22}$$

As we minimize $y_{q,d,t}$ then it will be equal to the right-hand side of (3.22) and we insert this in (3.20):

$$s_{q,d,t} \geq \frac{1}{2}\left(x_{q,d,t} - x_{q,d,t-1} - x_{q,d,t+1}\right) \tag{3.23}$$

Here we see that in the LP relaxation, the isolated lectures are only penalised in the disjunctive formulation by half of what they are penalised by the original formulation. The disjunctive formulation also resulted in a poorer performance than the original formulation when we tested

it in the commercial MIP solvers such as Gurobi by Gurobi Optimization Inc. (2015) and CPLEX by International Business Machines Corp. (2017). CPLEX provides the user with the capability of implementing custom branching decisions. So to get around the issues of the disjunctive formulation, we implemented a custom branching decision in CPLEX inspired by the disjunctive formulation. The idea is to provide CPLEX with the original formulation, i.e., with the formulation of the isolated lectures in (3.11). A common branching decision is to choose an integer variable x with a fractional value $\overline{x}$ in the LP relaxation and then apply the branching:

$$x \leq \lfloor \overline{x} \rfloor \quad \vee \quad x \geq \lceil \overline{x} \rceil \tag{3.24}$$

Instead of choosing one variable for the branching decision, we consider a sum $\overline{x}_{q,d,t} + \overline{x}_{q,d,t-1} + \overline{x}_{q,d,t+1}$. We then compute the penalty of the isolated lecture using the disjunctive formulation. If the value of the disjunctive formulation is greater than $s_{q,d,t}$ then the sum is a candidate for the branching decision, other we let CPLEX decide on the branching. If this sum is between one and two, then we apply the branching from the disjunction in (3.18). If the sum is between zero and one, then we apply the branching in (3.25). If the sum is equal to one, then we pick one of the two branching decisions randomly.

$$\left\{ x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} \leq 0 \right\} \quad \bigvee \quad \left\{ \begin{array}{l} x_{q,d,t} + x_{q,d,t-1} + x_{q,d,t+1} \geq 1 \\ s_{q,d,t} \geq 1 - x_{q,d,t-1} - x_{q,d,t+1} \end{array} \right\} \tag{3.25}$$

The question left to answer is which sum to choose. When branching on a single variables, then a common approach is to pick the variable which is most fractional, i.e., if $\lfloor \overline{x} \rceil$ is the value $\overline{x}$ rounded to the nearest integer then we pick the variable which maximizes $|\overline{x} - \lfloor \overline{x} \rceil|$. This branching rule means that we branch on the variable that violates the integrality requirement the most. We do something similar for the calculation of the isolated lectures. Consider curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$. Let $\overline{s}^{LP}_{q,d,t}$ be the value of the variable $s_{q,d,t}$ calculated in the optimal solution of the LP relaxation using the constraints (3.11) and $s_{q,d,t} \geq 0$. We calculate the sum $\overline{x}_{q,d,t} + \overline{x}_{q,d,t-1} + \overline{x}_{q,d,t+1}$. If the sum is less than or equal to one, then the disjunctive value $\overline{s}^{Disjunct}_{q,d,t}$ of the variable $s_{q,d,t}$ is set to $\overline{x}_{q,d,t}$. If the sum is greater than one and less than two, then we set $\overline{s}^{Disjunct}_{q,d,t} = 1 - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1}$. In all other cases we set $\overline{s}^{Disjunct}_{q,d,t} = 0$. Note that $\overline{s}^{Disjunct}_{q,d,t} \geq \overline{s}^{LP}_{q,d,t}$. We then pick the curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ which maximizes $\overline{s}^{viol} := \overline{s}^{Disjunct}_{q,d,t} - \overline{s}^{LP}_{q,d,t}$. If $\overline{s}^{viol} > 0$ we apply the branching (3.18) or (3.25) depending on the sum as mentioned earlier, otherwise we let CPLEX decide. The issue we encountered is that when implementing custom branching decisions in CPLEX a lot of the internal features is turned off. So the lower bounds were not as strong as without the custom branching, and the heuristics also did not produce solutions which are as good as the solutions obtained without the custom branching. Therefore, we suggest that any researchers that want to study this method further to consider using another framework such as SCIP (Gamrath et al., 2016) where the user is given more control of the Branch & Bound algorithm.

3.2 Lagrangian Relaxation

In this section we describe different methods that we have tested based on Lagrangian Relaxation. Before the description of the methods we provide an introduction to Lagrangian Relaxation in section 3.2.1. The idea of all the methods tested within this framework is to consider the formulation of the isolated lectures. In section 3.2.2 we apply the relaxation to the direct formulation of the isolated lectures. In section 3.2.3 we replace the formulation of the isolated lectures and describe how we apply Lagrangian Relaxation to this reformulation which results in a subproblem that is decomposable by the periods. In section 3.2.4 we reformulation the entire model such that the subproblem is decomposable by the curricula.

3.2.1 Introduction to Lagrangian Relaxation

In this section we provide a brief description of Lagrangian Relaxation similar to the description by Martin (1999, chapter 12), which we refer to for a detailed description of Lagrangian Relaxation. Consider a MIP problem in the following form:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b \\ & Bx \geq d \\ & x \in X \end{aligned} \tag{MIP}$$

The idea of Lagrangian Relaxation is to take a set of the constraints and *relax* them by multiplying them with a non-negative vector u, referred to as the *Lagrangian multipliers*, and inserting them in the objective function:

$$L(u) = \min \left\{ (c - B^\top u)^\top x + d^\top u \,|\, Ax \geq b, x \in X \right\} \tag{3.26}$$

We refer to the problem (3.26) as the *Lagrangian Subproblem*. For a fixed value of u the optimal solution of the subproblem provides a lower bound for (MIP). The goal is to maximize this lower bound by solving the following model:

$$\max \{ L(u) \,|\, u \geq 0 \} \tag{LR}$$

We let z_{MIP} and z_{LP} be the objective values of the optimal solution of (MIP) and the LP relaxation of the model respectively. Furthermore, we let z_{LR} be the objective value of the optimal solution to (LR), and we have the following relation

$$z_{LP} \leq z_{LR} \leq z_{MIP} \tag{3.27}$$

If the model (3.26) contains the *integrality property*, i.e., that the extreme points of the LP-relaxation are all integral, then $z_{LP} = z_{LR}$. So the set of constraints to relax should be selected such that the subproblem does not contain the *integrality property*. However, the constraints should also be selected such that the resulting subproblem is easier to solve than the original problem. Different methods can be applied to solve the problem (LR). One method is *Subgradient Search*. This algorithm is an iterative procedure which starts by setting the Lagrangian multipliers to some initial value, e.g., zero. Then the optimal solution of the subproblem (3.26) is found for these values of u. Let u^j be the value in the $j'th$ iteration of

the subgradient search and let x^j be the optimal solution to the subproblem for u^j. Then $d - Bx^j$ is a *subgradient* vector of the subproblem for u^j and for a scalar $\alpha_j > 0$ we calculate the multipliers for the next iteration as follows:

$$u^{j+1} = \max\left\{0, u^j + \alpha_j \left(d - Bx^j\right)\right\} \tag{3.28}$$

We let $\overline{L}$ be an upper bound on the value of the optimal solution to the Lagrangian Relaxation. This upper bound can for instance by the objective value of some feasible solution to the original model MIP. For a positive scalar θ_j between zero and two, the choice of α_j can be determined by:

$$\alpha_j = \frac{\theta_j \left(\overline{L} - L\left(u^j\right)\right)}{\|d - Bx^j\|^2} \tag{3.29}$$

The value of θ_j can for instance be set to two in the first iteration and then halved whenever the lower bound has not improved for some number of iterations. We refer to Martin (1999, section 12.5) for descriptions of other methods to solve the Lagrangian Relaxation.

3.2.2 Isolated Lectures

The idea in this section is to relax the constraints (3.11) using as it was shown in section 1.1 that the problem is much easier to solve without these constraints. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ we let $u_{q,d,t}$ be the non-negative Lagrangian multiplier of the associated constraint (3.11). The objective function of the Lagrangian Relaxation then becomes:

$$\begin{aligned}
&\sum_{c\in\mathcal{C},d\in\mathcal{D},t\in\mathcal{T},r\in\mathcal{R}} \left(W^{\mathbf{RC}} \left(S_x - C_r\right)^+ + \sum_{q\in\mathcal{Q}_c} \left(u_{q,d,t} - u_{q,d,t-1} - u_{q,d,t+1}\right) \right) x_{c,d,t,r} \\
&+ W^{\mathbf{RStab}} \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}} z_{c,r} - 1 \right) \\
&+ W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c \\
&+ \sum_{q\in\mathcal{Q},d\in\mathcal{D},t\in\mathcal{T}} \left(W^{\mathbf{IL}} - u_{q,d,t}\right) s_{q,d,t}
\end{aligned} \tag{3.30}$$

As the s variables do not contribute to any constraints in the Lagrangian Relaxation, then we can calculate the values of each of the variables in s independently. Consider a curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$. If $u_{q,d,t} < W^{\mathbf{IL}}$ then the coefficient of the $s_{q,d,t}$ is strictly positive and since we minimize this variable then the optimal value $\overline{s}_{q,d,t}$ is zero. If $u_{q,d,t} > W^{\mathbf{IL}}$ then the coefficient is strictly negative and the optimal value $\overline{s}_{q,d,t}$ is one. If $u_{q,d,t} = W^{\mathbf{IL}}$ then we can set $s_{q,d,t}$ to be either zero or one, and we set the value $\overline{s}_{q,d,t}$ to one if $\overline{x}_{q,d,t} - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} = 1$ and zero otherwise. The reason for these latter choices of $\overline{s}_{q,d,t}$ when $u_{q,d,t} = W^{\mathbf{IL}}$ is that the subgradient $\overline{x}_{q,d,t} - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} - \overline{s}_{q,d,t}$ then evaluates to zero. A summary of the values of $\overline{s}_{q,d,t}$ are provided in (3.31).

$$\overline{s}_{q,d,t} = \begin{cases} 1 & \text{if } u_{q,d,t} > W^{\mathbf{IL}} \vee \left(u_{q,d,t} = W^{\mathbf{IL}} \wedge \overline{x}_{q,d,t} - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} = 1\right) \\ 0 & \text{otherwise} \end{cases} \tag{3.31}$$

We then calculate the step size α for some positive scalar θ between zero and two as follows:

$$\alpha = \frac{\theta\left(\overline{L} - L\left(u\right)\right)}{\sum\limits_{q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}} \left(\overline{x}_{q,d,t} - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} - \overline{s}_{q,d,t}\right)^2} \tag{3.32}$$

Since the solution to the subproblem is feasible for the original model we can calculate the real objective value of the solutions and use them for the upper bound $\overline{L}$. Then for each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ we add $\alpha\left(\overline{x}_{q,d,t} - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} - \overline{s}_{q,d,t}\right)$ to $u_{q,d,t}$. As $u_{q,d,t}$ is a non-negative multiplier we set it to zero if it is negative after we updated it. Furthermore, we set $u_{q,d,t}$ to $W^{\mathbf{IL}}$ if the value is greater than $W^{\mathbf{IL}}$ after the update. To see the reason that we can set this upper bound on the multiplier, we reconsider the choice of the step size α. Another choice for α is a constant value ϵ. Assume that we choose ϵ to be infinitesimally small. Consider some iteration in the subgradient search and let there be one Lagrangian multiplier which is infinitesimally close to $W^{\mathbf{IL}}$ and where the gradient is positive. This means that when we update the multiplier it will be set to $W^{\mathbf{IL}}$ and for this value the gradient can never by positive, which means that the multiplier will never be greater than $W^{\mathbf{IL}}$. So we can use this value as an upper bound for the multipliers.

The issues encountered with our implementation of the Lagrangian Relaxation is that, though the subproblem is much easier to solve than the original model, the number of iterations required to find the optimal value of is large. Furthermore, the bounds obtained in the end of the subgradient search were not higher than what the solver Gurobi can obtain in the cutting plane phase of the root node in the original formulation.

3.2.3 Period Decomposition

The idea in this section is to two new binary variables for each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$; $x^-_{q,d,t}$ which is one if q has a lecture scheduled in the time slot before t on day d and $x^+_{q,d,t}$ which is one if q has a lecture scheduled in the time slot after t on day d. For ease of notation we define $x^-_{q,d,t}$ as zero if t is the first time slot and $x^+_{q,d,t}$ if t is the last time slot. We can then reformulate (3.11) into the following:

$$s_{q,d,t} \geq \sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} x_{c,d,t,r} - x^-_{q,d,t} - x^+_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.33}$$

We need to link the variables x^- and x^+ together with the x variables:

$$\sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} x_{c,d,t,r} = x^-_{q,d,t+1}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.34}$$

$$\sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} x_{c,d,t,r} = x^+_{q,d,t-1}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.35}$$

If we make a Lagrangian Relaxation where we multiply every constraint other than (3.3), (3.4) and (3.33) then the result is a subproblem which is decomposable on the periods. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$, time slot $t \in \mathcal{T}$ we let $\pi^-_{q,d,t}$ and $\pi^+_{q,d,t}$ be the Lagrangian multipliers of (3.34) and (3.35) respectively. For each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, time slot $t \in \mathcal{T}$ and room $r \in \mathcal{R}$ let $\pi_{c,d,t,r}$ be the sum of all the Lagrangian multipliers with respect to the contribution of $x_{c,d,t,r}$ to each of the relaxed constraints. π_0 is the sum of all the constants in the relaxed constraints, multiplied by the respective Lagrangian multipliers. The result is the following model for each day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$:

$$\begin{aligned}
\min \quad & W^{\mathbf{IL}} \sum_{q \in \mathcal{Q}} s_{q,d,t} + \sum_{c \in \mathcal{C}, r \in \mathcal{R}} \left((S_c - C_r)^+ + \pi_{c,d,t,r} \right) x_{c,d,t,r} \\
& + \sum_{q \in \mathcal{Q}} \pi^-_{q,d,t} x^-_{q,d,t} + \sum_{q \in \mathcal{Q}} \pi^+_{q,d,t} x^+_{q,d,t} + \pi_0 && && (3.36) \\
s.t. \quad & \sum_{c \in \mathcal{C}_\gamma, r \in \mathcal{R}} x_{c,d,t,r} && \leq 1, \quad && \forall \gamma \in \Gamma && (3.37) \\
& \sum_{c \in \mathcal{C}} x_{c,d,t,r} && \leq 1, \quad && \forall r \in \mathcal{R} && (3.38) \\
& \sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} x_{c,d,t,r} - x^-_{q,d,t} - x^+_{q,d,t} && \leq s_{q,d,t}, \quad && \forall q \in \mathcal{Q} && (3.39) \\
& x_{c,d,t,r} \in \mathbb{B}, && && \forall c \in \mathcal{C}, r \in \mathcal{R} && (3.40) \\
& x^-_{q,d,t} \in \mathbb{B}, x^+_{q,d,t} \in \mathbb{B}, s_{q,d,t} \in \mathbb{B}, && && \forall q \in \mathcal{Q} && (3.41)
\end{aligned}$$

The model (3.36) – (3.41) is much easier to solve than the original formulation. The issue is that the majority of the constraints are relaxed and the bounds we obtained in the Subgradient Search were not better then the bounds that Gurobi obtained in a shorter amount of time in the root node of the Branch & Bound tree. The model (3.36) – (3.41) can also be used in a *Column Generation* scheme. Cacchiani et al. (2013) provide an overview of different formulations to be incorporated in a column generation algorithm where one of them is similar to the one described here. They report results on four data instances and show that the bounds obtained by this formulation are lower and the running times are higher than for other formulations.

3.2.4 Curriculum Decomposition

In this section we describe a formulation based on making copies of some of the variables for each curriculum. Then we apply the Lagrangian Relaxation such that the model is decomposable into $|\mathcal{Q}| + 1$ subproblems. For each curriculum $q \in \mathcal{Q}$ we introduce the binary variable $x^q_{c,d,t}$ which is one if course $c \in \mathcal{C}_q$ has a lecture scheduled at day $d \in \mathcal{D}$ in time slot $t \in \mathcal{T}$, and zero otherwise. We then introduce the following constraints to link the variables together:

$$\sum_{r \in \mathcal{R}} x_{c,d,t,r} = x^q_{c,d,t}, \quad \forall q \in \mathcal{Q}, c \in \mathcal{C}_q, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.42}$$

The next step we perform is to take the variable $t_{c,d}$ for each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$, and make a copy of it for each curriculum $q \in \mathcal{Q}_c$. For each course $c \in \mathcal{C}$ we also make a copy of

the variable w_c for each curriculum $q \in \mathcal{Q}_c$. The constraints (3.8) and (3.9) are then replaced by constraints that use these new variables:

$$t^q_{c,d} - \sum_{t \in \mathcal{T}} x^q_{c,d,t} \leq 0, \qquad \forall q \in \mathcal{Q}, c \in \mathcal{C}_q, d \in \mathcal{D} \tag{3.43}$$

$$w^q_c + \sum_{d \in \mathcal{D}} t^q_{c,d} \quad \geq D^{\min}_c, \quad \forall q \in \mathcal{Q}, c \in \mathcal{C}_q \tag{3.44}$$

In the objective function (3.12) we apply the following substitution for each course $c \in \mathcal{C}$:

$$w_c = \frac{1}{|\mathcal{Q}_c|} \sum_{q \in \mathcal{Q}_c} w^q_c \tag{3.45}$$

We make a Lagrangian Relaxation by relaxing the linking constraints (3.42) for each curriculum $q \in \mathcal{Q}$, course $c \in \mathcal{C}_q$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ and we let $u^q_{c,d,t}$ be the Lagrangian multiplier. This makes the model decomposable such that for each curriculum $q \in \mathcal{Q}$ we have a subproblem LR_q containing all the variables associated with q, and we have a subproblem LR for the remaining variables. The subproblem LR is the same as the basic MIP formulation where the constraints (3.8), (3.9) and (3.11) are removed and objective function is replaced as follows:

$$W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} \left((S_c - C_r)^+ - \sum_{q \in \mathcal{Q}_q} u^q_{c,d,t} \right) x_{c,d,t,r} + W^{\mathbf{RStab}} \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) \tag{3.46}$$

For each curriculum $q \in \mathcal{Q}$ the subproblem LR_q is as follows:

$$\min \quad W^{\mathbf{IL}} \sum_{d \in \mathcal{D}, t\mathcal{T}} s_{q,d,t} + W^{\mathbf{MWD}} \sum_{c \in \mathcal{C}_q} \frac{1}{|\mathcal{Q}_c|} w^q_c + \sum_{c \in \mathcal{C}_q, d \in \mathcal{D}, t \in \mathcal{T}} u^q_{c,d,t} x^q_{c,d,t} \tag{3.47}$$

$$\text{s.t.} \quad \sum_{d \in \mathcal{D}, t \in \mathcal{T}} x^q_{c,d,t} = L_c \qquad \forall c \in \mathcal{C}_q \tag{3.48}$$

$$x^q_{c,d,t} \leq F_{c,d,t} \quad \forall c \in \mathcal{C}_q, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.49}$$

$$\sum_{c \in \mathcal{C}_q} x^q_{c,d,t} \leq 1 \qquad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{3.50}$$

$$t^q_{c,d} - \sum_{t \in \mathcal{T}} x^q_{c,d,t} \leq 0 \qquad \forall c \in \mathcal{C}_q, d \in \mathcal{D} \tag{3.51}$$

$$w^q_d + \sum_{d \in \mathcal{D}} t^q_{c,d} \geq D^{\min}_c \qquad \forall c \in \mathcal{C}_q \tag{3.52}$$

$$\sum_{c \in \mathcal{C}_q} \left(x^q_{c,d,t} - x^q_{c,d,t-1} - x^q_{c,d,t+1} \right) \leq s_{q,d,t} \qquad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{3.53}$$

$$x^q_{c,d,t} \in \mathbb{B} \qquad \forall c \in \mathcal{C}_q, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.54}$$

$$s_{q,d,t} \in \mathbb{B} \qquad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{3.55}$$

Note that in the subproblem LR_q we added the extra constraints (3.48), (3.49) and (3.50). The constraints (3.48) and (3.49) are copies of the constraints (3.1) and (3.2) and the constraints (3.50) comes from (3.3) by noting that the courses $\mathcal{C}_q$ constitute a course clique. The issue with this formulation is that the subgradient search requires many iterations to converge and for most of the data instances the bound obtained is not much higher than the bound calculated for the initial value (zero) of the Lagrangian multipliers. However, for the value of zero the lower bound obtained was at least as good as the bound by the LP-relaxation of the basic model (sometimes higher) and faster to compute. So we tested a Branch & Bound implementation where we used the Lagrangian Relaxation as the bounding problem instead of the LP-relaxation.

The issue arising here is how to apply the branching. As the LP-relaxation is a relaxation of the integrality requirements then it follows naturally to branch on fractional variables. In our Lagrangian Relaxation we relaxed the linking constraints (3.42) so we use them to apply the branching decision. If one of the constraints (3.42) is violated for some curriculum $q \in \mathcal{Q}$, course $c \in \mathcal{C}_q$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ then we apply the following branching:

$$\left\{ \begin{array}{l} x^q_{c,d,t} = 0,\ q \in \mathcal{Q}_c \\ \sum_{r \in \mathcal{R}} x_{c,d,t,r} = 0 \end{array} \right\} \quad \bigvee \quad \left\{ \begin{array}{l} x^q_{c,d,t} = 1,\ q \in \mathcal{Q}_c \\ \sum_{r \in \mathcal{R}} x_{c,d,t,r} = 1 \end{array} \right\} \tag{3.56}$$

The benefit of the branching (3.56) is that we do not need to resolve all the subproblems. For instance let course $c \in \mathcal{C}_q$, day $d \in \mathcal{D}$ and $t \in \mathcal{T}$ be the selected branching. Let $\overline{x}^q_{c,d,t}$ be the value of the variable $x^q_{c,d,t}$ that we branched on for each curriculum $q \in \mathcal{Q}$. For each curriculum $q \in \mathcal{Q}$ we do not need to resolve LR_q in the left branch if $\overline{x}^q_{c,d,t} = 0$ and we do not need to resolve LR_q in the right branch if $\overline{x}^q_{c,d,t} = 1$.

Another benefit of this approach is that the subproblem LR creates a solution which is feasible so we can calculate the value of the objective function for the basic model as an upper bound. However, this bound turned out to have a very high penalty of the constraints **MWD** and **IL**. To get around this issue we solved the LR subproblem in two stages. First we solved the model to obtain the optimal objective value $\overline{\beta}$ of LR. We then resolved the model where we added a constraint such that the objective value would not increase:

$$W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} \left((S_c - C_r)^+ - \sum_{q \in \mathcal{Q}_q} u^q_{c,d,t} \right) x_{c,d,t,r} + W^{\mathbf{RStab}} \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) \leq \overline{\beta} \tag{3.57}$$

For each curriculum $q \in \mathcal{Q}$, course $c \in \mathcal{C}_q$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ let $\overline{x}^q_{c,d,t}$ be the value of the variable $x^q_{c,d,t}$ in the optimal solution of the subproblem LR_q. We then defined the *distance* between the solution in LR to the solution in each subproblem LR_q for $q \in \mathcal{Q}$ as follows:

$$\sum_{c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} \left(\sum_{q \in \mathcal{Q}_c : \overline{x}^q_{c,d,t} = 0} x_{c,d,t,r} + \sum_{q \in \mathcal{Q}_c : \overline{x}^q_{c,d,t} = 1} (1 - x_{c,d,t,r}) \right) \tag{3.58}$$

The objective function was then replaced by minimizing the distance (3.58). The upper bounds obtained by Gurobis heuristic on the basic model is much better than the solutions

obtained in the subproblem LR. Furthermore, the lower bounds obtained by Gurobi are higher compared to the bounds produces by our approach.

3.3 Benders' Decomposition

In this section we describe different approaches based on Benders' Decomposition (Benders, 1962). We tested three applications of Benders' Decomposition. The first one is based on one of the flow formulations briefly mentioned in section 3.1 which are described in the paper in chapter 4. In chapter 4 it is shown that the problem has an underlying flow network. This network is projected out and replaced by constraints. We provide a brief description in section 2 and the details can be found in the paper in chapter 5. The second approach we tested is described in section 3.3.2, and is based on projecting out the formulations of all the soft constraints. The last approach we tested is described in section 3.3.3, and is based on a pattern formulation of the curricula. Before we present the applications we provide a brief introduction to Benders' Decomposition in section 3.3.1. This introduction is taken directly from the paper in chapter 5.

3.3.1 Introduction to Benders' Decomposition

Our introduction is a crude overview and we refer to (Benders, 1962) and Martin (1999, chapter 10) for a detailed description. We describe the method based on a model containing two types of variables, x and y. The x variables are non-negative continuous variables, and we do not have any assumptions on the y variables, i.e., $x \geq 0$ and $y \in Y$ where Y can be any domain, e.g., the set of integers. Consider the MIP model (3.59).

$$\begin{aligned} \min \quad & c^\top x + f(y) \\ \text{s.t.} \quad & Ax + B(y) \geq b \\ & y \in Y \\ & x \geq 0 \end{aligned} \tag{3.59}$$

In model (3.59) $c \in \mathbb{R}^n$ is the cost vector of the x variables, $A \in \mathbb{R}^{n \times m}$ is the constraint matrix of the x variables and $b \in \mathbb{R}^m$ is the right-hand-side vector of the constraints. $f : Y \to \mathbb{R}$ is some function to evaluate the cost of the y variables and B is a vector function that evaluates the contribution of the y variables for the constraints. If we fix the y variables to some value in the domain Y then what remains is a linear programme (LP). This assumption can be extended as described by Geoffrion (1972), but we stick to the (LP) case in this context. Model (3.59) can be rewritten to model (3.60).

$$\begin{aligned} \min \quad & f(y) + z \\ \text{s.t.} \quad & z \geq \min_{x \geq 0} \left\{ c^\top x \,|\, Ax \geq b - B(y) \right\} \\ & y \in Y \\ & z \in \mathbb{R} \end{aligned} \tag{3.60}$$

In model (3.60) there is an inner optimization problem in the constraints. If the y variables are fixed, then this is an LP and we can change it into its dual LP as in model (3.61).

$$
\begin{aligned}
\min \quad & f(y) + z \\
\text{s.t.} \quad & z \geq \max_{\pi \geq 0} \left\{ (b - B(y))\, \pi \mid A^\top \pi \leq c \right\} \\
& y \in Y \\
& z \in \mathbb{R}
\end{aligned}
\tag{3.61}
$$

One interesting aspect of the inner optimization problem in model (3.61) is that the corresponding polytope is independent of the values of the y variables. So if the polytope $\left\{ A^T \pi \leq c \right\}$ is non-empty then we can reformulate the problem using the extreme points Π^p and extreme rays Π^r as in model (3.62).

$$
\begin{aligned}
\min \quad & f(y) + z \\
\text{s.t.} \quad & z \geq (b - B(y))\, \overline{\pi}^p \quad \forall \overline{\pi}^p \in \Pi^p \\
& 0 \geq (b - B(y))\, \overline{\pi}^r \quad \forall \overline{\pi}^r \in \Pi^r \\
& y \in Y \\
& z \in \mathbb{R}
\end{aligned}
\tag{3.62}
$$

Model (3.62) is referred to as Benders' master problem. For a given solution $\overline{y}$ model (3.63) is referred to as Benders' subproblem.

$$
\begin{aligned}
\max \quad & (b - B(\overline{y}))\, \pi \\
\text{s.t.} \quad & A^\top \pi \leq c \\
& \pi \geq 0
\end{aligned}
\tag{3.63}
$$

As the number of extreme points and rays can be exponentially large, a way to solve the model is to relax the master problem by removing some (or all) of the constraints originating from the extreme points and rays and then iteratively add them as needed. This is done by finding a solution $\overline{y}$ for the master problem (3.62) and inserting the solution into the subproblem (3.63). The subproblem is then solved to obtain an extreme point $\overline{\pi}^p$ or ray $\overline{\pi}^r$ and the corresponding cut is added to the master problem if it is violated. This is done iteratively as illustrated in Figure 3.5.

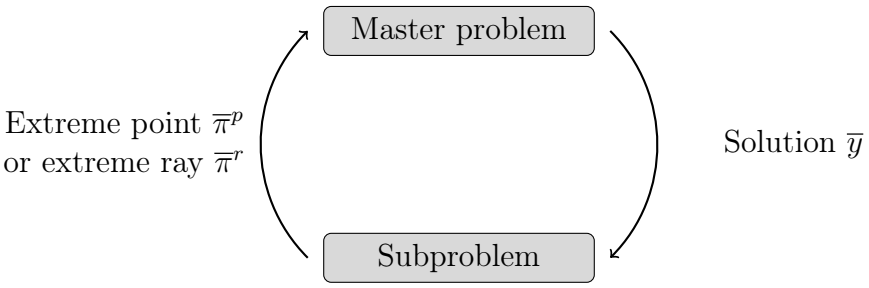


Figure 3.5: The iterative loop of Benders' algorithm.

A lower bound on the (relaxed) master problem is a lower bound on the original model, and if the subproblem returns an extreme point, then this provides an upper bound. The iterative process is run until some stopping criterion is met, e.g., a time limit is reached, or the lower and upper bounds are sufficiently close.

3.3.2 Penalty Variables

In this section we consider a reformulation of the basic model such that all the variables associated with the soft constraints can be relaxed to non-negative continuous variables. For each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ the variable $t_{c,d}$ is implicitly maximized. So if we consider an integer solution x then the upper bound of $t_{c,d}$ in constraint (3.8) will be either zero or greater than or equal to one. So if we add the upper bound $t_{c,d} \leq 1$ then we can relax the variable $t_{c,d}$ to be a non-negative continuous variable. We can also relax the variable w_c for each course $c \in \mathcal{C}$. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ the lower bound from constraint (3.11) for the variable $s_{q,d,t}$ is either one or less than or equal to zero, and as we are minimizing this variables then it can be relaxed to a non-negative continuous variable. We also relax the variable $z_{c,r}$ to a non-negative continuous variables for each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$. To be able to do this we replace the constraints (3.6) with the following constraints:

$$x_{c,d,t,r} \leq z_{c,r}, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R} \tag{3.64}$$

Consider a solution where $\overline{x}_{c,d,t,r}$ is the value of the variables $x_{c,d,t,r}$ for course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, time slot $t \in \mathcal{T}$ and room $r \in \mathcal{R}$. To obtain the objective value of the solution we solve the following LP model:

$$\begin{aligned}
\min \quad & W^{\mathbf{RStab}} \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) + W^{\mathbf{MWD}} \sum_{c \in \mathcal{C}} w_c + W^{\mathbf{IL}} \sum_{q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}} s_{q,d,t} \\
& + W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} (S_x - C_r)^+ \overline{x}_{c,d,t,r} && (3.65) \\
\text{s.t.} \quad & z_{c,r} \geq \overline{x}_{c,d,t,r}, && \forall c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R} && (3.66) \\
& w_c + \sum_{d \in \mathcal{D}} t_{c,d} \geq D_c^{\min}, && \forall c \in \mathcal{C} && (3.67) \\
& - t_{c,d} \geq - \sum_{t \in \mathcal{T}, r \in \mathcal{R}} \overline{x}_{c,d,t,r}, && \forall c \in \mathcal{C}, d \in \mathcal{D} && (3.68) \\
& - t_{c,d} \geq -1, && \forall c \in \mathcal{C}, d \in \mathcal{D} && (3.69) \\
& s_{q,d,t} \geq \sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} \left(\overline{x}_{c,d,t,r} - \overline{x}_{c,d,t-1,r} - \overline{x}_{c,d,t+1,r} \right), && \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} && (3.70) \\
& z_{c,r} \geq 0, && \forall c \in \mathcal{C}, r \in \mathcal{R} && (3.71) \\
& t_{c,d} \geq 0, && \forall c \in \mathcal{C}, d \in \mathcal{D} && (3.72) \\
& w_c \geq 0, && \forall c \in \mathcal{C} && (3.73) \\
& s_{q,d,t} \geq 0, && \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} && (3.74)
\end{aligned}$$

We assume that the model (3.65) – (3.74) is feasible for any solution $\overline{x}$. We let $\alpha_{c,d,t,r}$ for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, time slot $t \in \mathcal{T}$ and room $r \in \mathcal{R}$ be the dual variable of the corresponding constraint (3.66). For the constraints (3.67) we let π_c be the dual variable for each course $c \in \mathcal{C}$. For each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ we let $\beta_{c,d}$ and $\zeta_{c,d}$ be the dual variables of the constraints (3.68) and (3.69) respectively. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ we let $\mu_{q,d,t}$ be the dual variable of the constraint (3.70). We then formulate the dual of (3.65) – (3.74) as follows:

$$\max \sum_{c\in\mathcal{C},d\in\mathcal{D},t\in\mathcal{T},r\in\mathcal{R}} \overline{x}_{c,d,t,r}\alpha_{c,d,t,r} + \sum_{c\in\mathcal{C}} D_c^{\min}\pi_c - \sum_{c\in\mathcal{C},d\in\mathcal{D}} \left(\sum_{t\in\mathcal{T},r\in\mathcal{R}} \overline{x}_{c,d,t,r}\right)\beta_{c,d} - \sum_{c\in\mathcal{C},d\in\mathcal{D}} \zeta_{c,d}$$
$$+ \sum_{q\in\mathcal{Q},d\in\mathcal{D},t\in\mathcal{T}} \left(\sum_{c\in\mathcal{Q}_q,r\in\mathcal{R}} \left(\overline{x}_{c,d,t,r} - \overline{x}_{c,d,t-1,r} - \overline{x}_{c,d,t+1,r}\right)\right)\mu_{q,d,t}$$
$$+ W^{\mathbf{RC}} \sum_{c\in\mathcal{C},d\in\mathcal{D},t\in\mathcal{T},r\in\mathcal{R}} (S_x - C_r)^+ \overline{x}_{c,d,t,r} - W^{\mathbf{RStab}}|\mathcal{C}| \tag{3.75}$$

$$\text{s.t.} \sum_{d\in\mathcal{D},t\in\mathcal{T}} \alpha_{c,d,t,r} \le W^{\mathbf{RStab}}, \quad \forall c\in\mathcal{C}, r\in\mathcal{R} \tag{3.76}$$

$$\pi_c - \beta_{c,d} - \zeta_{c,d} \le 0, \quad \forall c\in\mathcal{C}, d\in\mathcal{D} \tag{3.77}$$

$$0 \le \pi_c \le W^{\mathbf{MWD}}, \quad \forall c\in\mathcal{C} \tag{3.78}$$

$$0 \le \mu_{q,d,t} \le W^{\mathbf{IL}}, \quad \forall q\in\mathcal{Q}, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.79}$$

$$\alpha_{c,d,t,r} \ge 0, \quad \forall c\in\mathcal{C}, d\in\mathcal{D}, t\in\mathcal{T}, r\in\mathcal{R} \tag{3.80}$$

$$\beta_{c,d} \ge 0, \quad \forall c\in\mathcal{C}, d\in\mathcal{D} \tag{3.81}$$

We add the continuous variable u to the model. We let the variable be non-negative as a trivial lower bound of CTT is zero. Then for a solution $(\overline{x}, \overline{u})$ we solve the dual LP model (3.75) – (3.81). If the objective value of the dual LP is greater than $\overline{u}$ then we add the following violated cut to the model:

$$u \ge \sum_{c\in\mathcal{C},d\in\mathcal{D},t\in\mathcal{T},r\in\mathcal{R}} \left(W^{\mathbf{RC}}(S_c - C_r)^+ + \overline{\alpha}_{c,d,t,r} - \overline{\beta}_{c,d} + \sum_{q\in\mathcal{Q}_c}\left(\overline{\mu}_{q,d,t} - \overline{\mu}_{q,d,t-1} - \overline{\mu}_{q,d,t+1}\right)\right) x_{c,d,t,r}$$
$$+ \sum_{c\in\mathcal{C}} D_c^{\min}\overline{\pi}_c - \sum_{c\in\mathcal{C},d\in\mathcal{D}} \overline{\zeta}_{c,d} - W^{\mathbf{RStab}}|\mathcal{C}| \tag{3.82}$$

The benefit of the approach described in this section that the model (3.75) – (3.81) can be solved analytically, thus we do not need to use an LP solver.

Consider a curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$. In model (3.75) – (3.81) the variable $\mu_{q,d,t}$ is not included in any constraints other than its own lower and upper bounds (3.80). So in the optimal solution we set the variable to its upper bound when the objective coefficient is positive, otherwise we set it to its lower bound:

$$\overline{\mu}_{q,d,t} = \begin{cases} W^{\mathbf{IL}}, & \text{if } \sum_{c\in\mathcal{C}_q,r\in\mathcal{R}} \left(\overline{x}_{c,d,t,r} - \overline{x}_{c,d,t-1,r} - \overline{x}_{c,d,t+1,r}\right) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.83}$$

Consider now course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$. We define $\overline{x}_{c,r}^{\max}$ as the maximum amount that c has been assigned to r in any period:

$$\overline{x}_{c,r}^{\max} := \max_{d \in \mathcal{D}, t \in \mathcal{T}} \{\overline{x}_{c,d,t,r}\} \tag{3.84}$$

In the constraint (3.76) the variables $\alpha_{c,d,t,r}$ for each day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ contribute by the same amount and they do not contribute to any other constraints. So we can pick any variable $\alpha_{c,d,t,r}$ where $\overline{x}_{c,d,t,r} = \overline{x}_{c,r}^{\max}$ and set it to $W^{\mathbf{RStab}}$ and the others to zero. Another approach is to distribute $W^{\mathbf{RStab}}$ across all the variables that have the largest coefficient. We define the set $\mathcal{P}^{\max} \subseteq \mathcal{D} \times \mathcal{T}$ as the set of periods where c has been assigned to r by the maximum amount:

$$\mathcal{P}^{\max} := \left\{ d \in \mathcal{D}, t \in \mathcal{T} \,|\, \overline{x}_{c,d,t,r} = \overline{x}_{c,r}^{\max} \right\} \tag{3.85}$$

We then calculate the value $\overline{\alpha}_{c,d,t,r}$ according to whether or not $(d,t) \in \mathcal{P}^{\max}$:

$$\overline{\alpha}_{c,d,t,r} = \begin{cases} \dfrac{W^{\mathbf{RStab}}}{|\mathcal{P}^{\max}|}, & \text{if } (d,t) \in \mathcal{P}^{\max} \\ 0, & \text{otherwise} \end{cases} \tag{3.86}$$

The last variables we need to calculate are the ones associated with the calculation of the **MWD** constraint. Note that the variables π, β and ζ can be calculated separately for each course. Consider course $c \in \mathcal{C}$ and assume that we know the optimal value $\overline{\pi}_c$ for the variable π_c. Due to constraints (3.77) we have that for the values $\overline{\beta}_{c,d}$ and $\overline{\zeta}_{c,d}$ for each day $d \in \mathcal{D}$ the sum $\overline{\beta}_{c,d} + \overline{\zeta}_{c,d}$ is lower bounded by $\overline{\pi}_c$. Since all the variables are non-negative and the coefficients of the variables $\beta_{c,d}$ and $\zeta_{c,d}$ for each day $d \in \mathcal{D}$ are non-positive then it is always desirable to have the sum at its lower bound:

$$\overline{\pi}_c = \overline{\beta}_{c,d} + \overline{\zeta}_{c,d} \tag{3.87}$$

As the variables $\beta_{c,d}$ and $\zeta_{c,d}$ do not contribute to other constraints then the values $\overline{\beta}_{c,d}$ and $\overline{\zeta}_{c,d}$ can be set such that the variable with the largest (least negative) coefficient is set to $\overline{\pi}_c$ and the other is set to zero:

$$\overline{\beta}_{c,d} = \begin{cases} \overline{\pi}_c, & \text{if } \displaystyle\sum_{t \in \mathcal{T}, r \in \mathcal{R}} \overline{x}_{c,d,t,r} \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.88}$$

$$\overline{\zeta}_{c,d} = \begin{cases} \overline{\pi}_c, & \text{if } \displaystyle\sum_{t \in \mathcal{T}, r \in \mathcal{R}} \overline{x}_{c,d,t,r} > 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.89}$$

If we insert (3.88) and (3.89) into the objective function then the objective coefficient of π_c becomes the following:

$$D_c^{\min} - \sum_{d \in \mathcal{D}} \min \left\{ 1, \sum_{d \in \mathcal{D}, t \in \mathcal{T}} \overline{x}_{c,d,t,r} \right\} \tag{3.90}$$

So we calculate (3.90) and it is positive then we set π_c to its upper bound $W^{\mathbf{MWD}}$ and then we use (3.88) and (3.89) to calculate the values of the β and ζ variables.

Every integer solution $\overline{x}$ is a feasible solution and provides an upper bound. We implemented the cuts in a callback function of Gurobi where we calculated the cuts analytically. Unfortunately, all the lower bounds were higher and the upper bounds were lower when the basic model is solved in Gurobi. Some of these issues might arise because Gurobi does not contain any information about the objective function that can guide the heuristics and the cutting planes. In the newest version of CPLEX an automatic Benders' Decomposition has been implemented. Future research could see if this implementation helps on the issues addressed here. Furthermore, it could also be interesting to include more soft constraints into the approach.

3.3.3 Curriculum Patterns

The idea in this section is to remove the s variables and then add a pattern variables instead. For each curriculum and day we construct a set of patterns that the curriculum can be scheduled to on that day. A pattern defines in which time slots that lectures are scheduled. For instance if $|\mathcal{T}| = 4$ then all the possible patterns that exists are illustrated in Table 3.1. Each column in the table corresponds to a pattern and each row corresponds to a time slot. If a pattern has a lecture scheduled in a specific time slot then this is marked with $\times$. The second last line (IL) count the number of isolated lectures that are created if a curriculum is assigned to the corresponding pattern on any day.

Table 3.1: The patterns when $|\mathcal{T}| = 4$.

time slot	pattern index k															
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×				×	×	×				×	×	×		×
2			×			×			×	×		×	×		×	×
3				×			×		×		×	×		×	×	×
4					×			×		×	×		×	×	×	×
IL	0	1	1	1	1	0	2	2	0	2	0	0	1	1	0	0

For each curriculum $q \in \mathcal{Q}$ and day $d \in \mathcal{D}$ we let $\mathcal{K}_{q,d}$ be the set of feasible patterns that q can be assigned to on day d. We define $\lambda^k_{q,d}$ as a binary variables which takes value one if the curriculum $q \in \mathcal{Q}$ is assigned to the pattern $k \in \mathcal{K}_{q,d}$ for day $d \in \mathcal{D}$. We link the x variables from the basic formulation from section 3.1 with the λ variables as follows:

$$\sum_{k \in \mathcal{K}_{q,d}} a^k_t \lambda^k_{q,d} = \sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} x_{c,d,t,r}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.91}$$

$$\sum_{k \in \mathcal{K}_{q,d}} \lambda^k_{q,d} = 1, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D} \tag{3.92}$$

$$\lambda^k_{q,d} \in \mathbb{B}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, k \in \mathcal{K}_{q,d} \tag{3.93}$$

Constraints (3.91) ensure that if a curriculum is assigned some pattern on a day, then all the courses in the curriculum must be assigned to the periods associated with the pattern. Constraints (3.92) ensure that every curriculum selects exactly one pattern for each day. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{q,d}$ we let $\alpha^k_{q,d}$ be the total penalty of the

pattern k, thus we add $\alpha^k_{q,d}\lambda^k_{q,d}$ to the objective function. Consider some solution where each variable $x_{c,d,t,r}$ is fixed to some integer value $\overline{x}_{c,d,t,r}$ for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, time slot $t \in \mathcal{T}$ and room $r \in \mathcal{R}$. Then the remaining problem to solve is the following:

$$\min \quad \sum_{q\in\mathcal{Q},d\in\mathcal{D},k\in\mathcal{K}_{q,d}} \alpha^k_{q,d}\lambda^k_{c,d} \tag{3.94}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}_{q,d}} a^k_t\lambda^k_{q,d} = \sum_{c\in\mathcal{C}_q,r\in\mathcal{R}} \overline{x}_{c,d,t,r}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.95}$$

$$\sum_{k\in\mathcal{K}_{q,d}} \lambda^k_{q,d} = 1, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D} \tag{3.96}$$

$$\lambda^k_{q,d} \in \mathbb{B}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, k \in \mathcal{K}_{q,d} \tag{3.97}$$

Assume that model (3.94) – (3.97) is feasible and bounded. Furthermore, assume that all the extreme points of the polytope of the LP-relaxation of model (3.94) – (3.97) are integer points, meaning that we can replace the integrality requirements (3.97) by the following non-negativity constraints:

$$\lambda^k_{q,d} \in \geq 0, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, k \in \mathcal{K}_{q,d} \tag{3.98}$$

Then we can introduce the continuous variable z and reformulate model (3.94) – (3.97) as follows:

$$\min \quad z \tag{3.99}$$

$$\text{s.t.} \quad z \geq \sum_{c\in\mathcal{C},d\in\mathcal{D},t\in\mathcal{T},r\in\mathcal{R}} \pi^j_{c,d,t,r}\overline{x}_{c,d,t,r} + \pi^j_0, \quad \forall j \in \mathcal{J} \tag{3.100}$$

$\mathcal{J}$ is the set of Benders' optimality cuts and for each cut $j \in \mathcal{J}$ the coefficient of the variable $x_{c,d,t,r}$ for course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, time slot $t \in \mathcal{T}$ and room $r \in \mathcal{R}$ is $\pi^j_{c,d,t,r}$ and π^j_0 is a constant. We add z to the objective function of the original formulation and all the λ variables from the model. As $\mathcal{J}$ is exponentially large, we remove them all (except the trivial $z \geq 0$) and add them dynamically when they are violated. We define μ as the dual vector of the constraints (3.95) and π as the dual vector of (3.96). We formulate the dual of model (3.94) – (3.97) as follows:

$$\max \quad \sum_{q\in\mathcal{Q},d\in\mathcal{D},t\in\mathcal{T}} \left(\sum_{c\in\mathcal{C}_q,r\in\mathcal{R}} \overline{x}_{c,d,t,r} \right) \mu_{q,d,t} + \sum_{q\in\mathcal{Q},d\in\mathcal{D}} \pi_{q,d} \tag{3.101}$$

$$\text{s.t.} \quad \mu_{q,d,t} + \pi_{q,d} \leq \alpha^k_{q,d}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, k \in \mathcal{K}_{q,d} \tag{3.102}$$

$$\mu_{q,d,t} \in \mathbb{R}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.103}$$

$$\pi_{q,d} \in \mathbb{R}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D} \tag{3.104}$$

Given an integer solution $(\overline{x}, \overline{z})$ we solve model (3.101) – (3.104) and get the optimal solution $(\overline{\mu}, \overline{\pi})$. If the objective value of $(\overline{\mu}, \overline{\pi})$ is greater than $\overline{z}$ then we have a violated optimality cut which we add to the model:

$$z \geq \sum_{c \in \mathcal{C}, d \in \mathcal{D}, t \in \mathcal{T}, r \in \mathcal{R}} \left(\sum_{q \in \mathcal{Q}_c} \overline{\mu}_{q,d,t} \right) x_{c,d,t,r} + \sum_{q \in \mathcal{Q}, d \in \mathcal{D}} \overline{\pi}_{q,d} \tag{3.105}$$

These cuts were based on the assumption that model (3.94) – (3.97) has the integrality property, i.e., that all the extreme points of the polytope of the LP-relaxation are integral. Consider a solution $\overline{x}$ for the variables x and the polytope of LP relaxation of (3.91) – (3.93). If $\sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} \overline{x}_{c,d,t,r} = 0$ for curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ then for each pattern $k \in \mathcal{K}_{q,d}$ where $a_t^k = 1$ the variable $\lambda_{q,d}^k$ must be zero in any solution so we remove this variable. When we have removed all the variables that contribute to the constraint then we remove the constraint as it is now redundant. What remains is a zero–one *set partitioning problem*. Note that the model is decomposable for each curriculum and each day, i.e., for each curriculum $q \in \mathcal{Q}$ and each day $d \in \mathcal{D}$ we consider the following problem:

$$\sum_{k \in \mathcal{K}_{q,d}} a_t^k \lambda_{q,d}^k = 1, \quad \forall t \in \mathcal{T} : \sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} \overline{x}_{c,d,t,r} = 1 \tag{3.106}$$

$$\sum_{k \in \mathcal{K}_{q,d}} \lambda_{q,d}^k = 1 \tag{3.107}$$

$$\lambda_{q,d}^k \geq 0, \qquad \forall k \in \mathcal{K}_{q,d} : a_t^k = \sum_{c \in \mathcal{C}_q, r \in \mathcal{R}} \overline{x}_{c,d,t,r}, \forall t \in \mathcal{T} \tag{3.108}$$

If model (3.106) – (3.108) has the integrality property for each curriculum and each day then model (3.94) – (3.97) also has the integrality property. A zero–one set-partitioning problem is said to have the integrality property if and only if the corresponding constraint matrix is *perfect* (Padberg, 1974; Ryan and Falkner, 1988). So we need to show that the constraint matrix of model (3.106) – (3.108) is *perfect*. To show that the constraint matrix is perfect we use definition 3.1 and theorem 3.1.

Definition 3.1 (Padberg (1974) and Ryan and Falkner (1988)). *An $m \times h$ zero–one matrix A_h with $h \leq m$ is said to have the property $\Pi_{\beta,h}$ if:*

- *A_h contains a $h \times h$ non-singular submatrix B_h were all rows and columns sum to β*
- *Every row in A_h which is not in B_h is either equal to a row in B_h or the sum of the row is strictly less than β*

Theorem 3.1 (Padberg (1974) and Ryan and Falkner (1988)). *Any zero–one matrix A of size $m \times n$ is* perfect *if and only if A does not contain any $m \times h$ submatrix A_h with the property $\Pi_{\beta,h}$ for $\beta \geq 2$ and $3 \leq h \leq m$.*

We let A of size $m \times n$ be the constraint matrix of model (3.106) – (3.108). To prove that A is perfect we assume that it is not perfect and show that this leads to a contradiction. The assumption means that there is an $m \times h$ submatrix A_h of A with property $\Pi_{\beta,h}$ for some $\beta \geq 2$ and $3 \leq h \leq m$. As A_h has the property $\Pi_{\beta,h}$ then it contains a $h \times h$ non-singular submatrix B_h where all rows and columns sum to β. Since the number of columns in B_h is h then an

upper bound on β is h. If $\beta = h$ then B_h cannot be non-singular, i.e., $\beta \leq h - 1$ must hold. Consider the row of A_h corresponding to constraint (3.107). As the coefficients of the variables are all one in this constraint, then this means that the sum of this row is h. As $\beta < h$ then the row cannot be in B_h and thus (3.107) must be one of the rows in A_h which is not in B_h. However, this contradicts definition 3.1 as the sum of any row in A_h outside B_h must be either equal to a row in B_h or sum to a value less than β. So the constraint matrix of model (3.106) – (3.108) must be perfect.

The issue we encountered with this approach is that Gurobi had difficulties improving the lower bounds compared to the basic formulation. We only added the Benders' cuts for integer solutions. Future research could consider to add the cuts in fractional solutions as well. However, the model that generates the cuts are based on the property that it is feasible and bounded for integer solution. This does not always apply to fractional solutions, so feasibility cuts should also be considered. Future research could also consider other hard or soft constraints that can be included in this formulation. For instance if we penalize the minimum or maximum number of lectures it is allowed to schedule for each curriculum. This penalty can be added to the costs of the patterns or if it is a hard constraint then we can remove the patterns that violates this. Another example of a penalty that can be included is *windows*, i.e., if a curriculum has more than one lecture scheduled in one day then they should be scheduled consecutively and any *holes* are penalized. The benefit of the pattern formulation is that the penalties of the mentioned soft constraints can be based on non-linear expressions.

3.4 Cutting Planes

In this section we provide three different cutting plane methods that we have tested. Before the descriptions of the cutting plane approaches we provide an introduction to cutting plane methods in section 3.4.1. In section 3.4.2 we describe a cutting plane method based on a pattern formulation and Benders' Decomposition. The last two methods are disjunctive cuts where the first approach in section 3.4.3 is based on the disjunctive formulation described in section 3.1.1 and the last approach in section 3.4.4 is based on a non-linear formulation of the isolated lectures.

3.4.1 Introduction to Cutting Planes

In this section we provide a brief description of Cutting Plane algorithms. We refer to Wolsey (1998) for a thorough description. Consider a MIP problem in the following form:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \geq b \\ & x \in X \end{aligned} \tag{MIP}$$

where c, x and b are vectors and A is a matrix and X is some set, e.g. the set of integers. To get a lower bound of the model (MIP) the LP relaxation is solved. Let $P_{IP} = conv\left(\{Ax \leq b, x \in X\}\right)$ and let $P_{LP} = \{Ax \leq b, x \in \mathbb{R}^n\}$, i.e., P_{IP} is the convex hull of the model (MIP) and P_{LP} is the LP relaxation; $P_{LP} \supseteq P_{IP}$. The idea of cutting plane algorithms is to solve the LP relaxation to obtain a point $x^* \in P_{LP}$. If $x^* \notin P_{IP}$ then we look for an

inequality $\pi^\top c \leq \pi_0$ which is violated by x^* but not violated by any point $x \in P_{IP}$. The idea is illustrated in Figure 3.6.

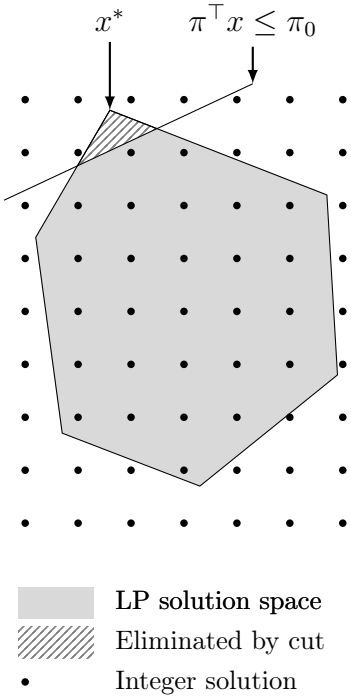


Figure 3.6: Illustration of the solution space.

In Figure 3.6 an example of the solution space is illustrated, where each point corresponds to an integer solution. When the LP relaxation is solved to optimality the point x^* is obtained. This solution is not integer and so an inequality $\pi^\top x \leq \pi_0$ is added that separates x^* from the set of integer solutions. This approach can be iterated until no more cuts can be found, which results in the polyhedron P' where $P_{LP} \supseteq P' \supseteq P_{IP}$, and hopefully $P_{LP} \supset P'$. $P' \cap X$ can then be explored by a Branch & Bound algorithm to find the optimal solution. If P' is significantly smaller than P_{LP} then this can improve the performance of the Branch & Bound algorithm.

3.4.2 Clique Pattern Cuts

The idea in this section is to make a formulation similar to the pattern formulation in section 3.3.3 and the formulation in the paper in chapter 6. However, instead of considering patterns for one course, we consider patterns for an entire course clique.

For each course clique $\gamma \in \Gamma$ and day $d \in \mathcal{D}$ we let $\mathcal{K}_{\gamma,d}$ be the set of feasible patterns that γ can be assigned to on day d similar to section 3.3.3. We define $\lambda^k_{\gamma,d}$ as a binary variables which takes value one if the clique $\gamma \in \Gamma$ is assigned the pattern $k \in \mathcal{K}_{\gamma,d}$ for day $d \in \mathcal{D}$. Then we can link the x variables from the compact formulation from section 3.1 together with the λ variables as follows:

$$\sum_{k\in\mathcal{K}_{\gamma,d}} a_t^k \lambda_{\gamma,d}^k = \sum_{c\in\mathcal{C}_\gamma, r\in\mathcal{R}} x_{c,d,t,r}, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.109}$$

$$\sum_{k\in\mathcal{K}_{\gamma,d}} \lambda_{\gamma,d}^k = 1, \quad \forall \gamma\in\Gamma, d\in\mathcal{D} \tag{3.110}$$

$$\lambda_{\gamma,d}^k \in \mathbb{B}, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, k\in\mathcal{K}_{\gamma,d} \tag{3.111}$$

We can prove that the integrality requirements can be replaced by non-negativity constraint by using the same arguments as in section 3.3.3 where we replace the curricula by the course cliques. The difference between the approach in section 3.3.3 and the approach we consider here is that we use the the model to generate cuts for fractional solutions. If we are provided with a fractional solution $\overline{x}$ then we chec whether the LP-relaxation of model (3.109) – (3.110) is infeasible. If this is the case then we can generate a cut. We do this by introducing new variables. For each clique $\gamma\in\Gamma$, day $d\in\mathcal{D}$ and time slot $t\in\mathcal{T}$ we let the variables $u_{\gamma,d,t}^+$ and $u_{\gamma,d,t}^-$ be the violation of the constraint (3.109). For each clique $\gamma\in\Gamma$ and day $d\in\mathcal{D}$ we let the variables $v_{\gamma,d}^+$ and $v_{\gamma,d}^-$ be the violation of the constraint (3.110). The objective is to minimize these variables, i.e., we solve the following model:

$$\min \quad \sum_{\gamma\in\Gamma,d\in\mathcal{D},t\in\mathcal{T}} \left(u_{\gamma,d,t}^+ + u_{\gamma,d,t}^-\right) + \sum_{\gamma\in\Gamma,d\in\mathcal{D}} \left(v_{\gamma,d}^+ + v_{\gamma,d}^-\right) \tag{3.112}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}_{\gamma,d}} a_t^k \lambda_{\gamma,d}^k + u_{\gamma,d,t}^+ - u_{\gamma,d,t}^- = \sum_{c\in\mathcal{C}_\gamma, r\in\mathcal{R}} \overline{x}_{c,d,t,r}, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.113}$$

$$\sum_{k\in\mathcal{K}_{\gamma,d}} \lambda_{\gamma,d}^k + v_{\gamma,d}^+ - v_{\gamma,d}^- = 1, \quad \forall \gamma\in\Gamma, d\in\mathcal{D} \tag{3.114}$$

$$u_{\gamma,d,t}^+, u_{\gamma,d,t}^- \geq 0 \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.115}$$

$$v_{\gamma,d}^+, v_{\gamma,d}^- \geq 0 \quad \forall \gamma\in\Gamma, d\in\mathcal{D} \tag{3.116}$$

$$\lambda_{\gamma,d}^k \geq 0, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, k\in\mathcal{K}_{\gamma,d} \tag{3.117}$$

If the objective value of the optimal solution of model (3.112) – (3.117) is greater than zero then we can generate a cut to remove the fractional solution $\overline{x}$ from the solution space of the basic model. The cut is generated by dualizing model (3.112) – (3.117) as follows:

$$\max \quad \sum_{\gamma\in\Gamma,d\in\mathcal{D},t\in\mathcal{T}} \left(\sum_{c\in\mathcal{C}_\gamma, r\in\mathcal{R}} \overline{x}_{c,d,t,r}\right) \alpha_{\gamma,d,t} + \sum_{\gamma\in\Gamma,d\in\mathcal{D}} \pi_{\gamma,d} \tag{3.118}$$

$$\text{s.t.} \quad \alpha_{\gamma,d,t} + \pi_{\gamma,d} \leq 0, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, k\in\mathcal{K}_{\gamma,d} \tag{3.119}$$

$$-1 \leq \alpha_{\gamma,d,t} \leq 1, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.120}$$

$$-1 \leq \pi_{\gamma,d} \leq 1, \quad \forall \gamma\in\Gamma, d\in\mathcal{D} \tag{3.121}$$

$$\alpha_{\gamma,d,t} \in \mathbb{R}, \quad \forall \gamma\in\Gamma, d\in\mathcal{D}, t\in\mathcal{T} \tag{3.122}$$

$$\pi_{\gamma,d} \in \mathbb{R}, \quad \forall \gamma\in\Gamma, d\in\mathcal{D} \tag{3.123}$$

We solve the model (3.118) – (3.123) and obtain the optimal solution $(\overline{\alpha}, \overline{\pi})$. If the objective value of this solution is strictly positive then we add the following cut to the basic model:

$$\sum_{c\in\mathcal{C},d\in\mathcal{D},t\in\mathcal{T}} \left(\sum_{\gamma\in\Gamma_c} \overline{\alpha}_{\gamma,d,t} \right) x_{c,d,t} + \sum_{\gamma\in\Gamma,d\in\mathcal{D}} \overline{\pi}_{\gamma,d} \leq 0 \tag{3.124}$$

To improve the strength of the cuts, we also apply a preprocessing phase to remove patterns from the set $\mathcal{K}_{\gamma,d}$ for each clique $\gamma \in \Gamma$ and day $d \in \mathcal{D}$. The preprocessing we apply are taken from the paper in chapter 6. All the preprocessing techniques that are discussed in the paper can be applied by extending them to consider course cliques instead of only single courses. Some of the inequalities discussed in the paper can also be applied here, such as the *extended cover inequalities*. Furthermore, consider two cliques $\gamma_1 \in \Gamma$ and $\gamma_2 \in \Gamma$. Let $\mathcal{C}_{\gamma_1,d} \subseteq \mathcal{C}_{\gamma_1}$ for γ_1 be the set of courses where at least one of the time slots on day $d \in \mathcal{D}$ is available and similarly for γ_2 we have the set $\mathcal{C}_{\gamma_2,d} \subseteq \mathcal{C}_{\gamma_2}$. If $\mathcal{C}_{\gamma_1,d} = \mathcal{C}_{\gamma_2,d}$ then we can remove all the constraints corresponding to clique γ_2 and day $d \in \mathcal{D}$ as they are redundant.

We implemented the cuts in a callback function of Gurobi where we added any violated cut to the fractional solution generated by Gurobi in the root node of the Branch & Bound tree. The cuts did not improve the lower bounds compared to Gurobi's default behaviour. Any future research focusing on this approach should consider if additional valid inequalities can be derived for model (3.112) – (3.117) to strengthen the cuts. It could also be interesting to see if MIP models for other scheduling problems would benefit from these cuts.

3.4.3 Disjunctive Cuts

In this section we describe an application of adding *disjunctive cuts* to the basic model. First we provide a brief description of disjunctive cuts, and we refer to Fischetti et al. (2011) for a thorough description.

We consider the following MIP model:

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in X \end{aligned} \tag{3.125}$$

Let P be the polyhedron of the LP-relaxation of model (3.125), i.e., $P = \{Ax \leq b, x \in \mathbb{R}^n\}$ where n is the number of variables in the vector x. Assume that we are given a disjunction $\left(P^L, P^R\right)$ of the model such that every integer point in P is in P^R or P^L but not in both of them. We let P^L be denoted as $\left\{A^L x \leq b^L, x \in \mathbb{R}\right\}$ and we let P^R be denoted as $\left\{A^R x \leq b^R, x \in \mathbb{R}^n\right\}$.

For any dual vector $u \geq 0$ the cut $(uA^L)^\top x \leq ub^L$ is valid for P^L and for any dual vector $v \geq 0$ the cut $(vA^L)^\top x \leq vb^L$ is valid for P^R. We let A^L_i be the column in the matrix A^L corresponding to the variable x_i and similarly for A^R_i. We can use these two cuts to generate a valid inequality $\pi^\top x \leq \pi_0$ for P by calculating π and π_0 as follows:

$$\pi_i = \min\left\{uA^L_i, vA^L_i\right\}, \quad i \in \{1, \ldots, n\} \tag{3.126}$$

$$\pi_0 = \max\left\{ub^L, vb^R\right\} \tag{3.127}$$

Consider a fractional solution $\overline{x}$ for the LP relaxation of (3.125). We then formulate the following cut separation problem:

$$\begin{aligned}
\max \quad & \pi^\top \overline{x} - \pi_0 & \\
\text{s.t.} \quad & \pi & \leq uA^L \\
& \pi & \leq vA^R \\
& \pi_0 & \geq ub^L \\
& \pi_0 & \geq vb^R \\
& e^\top u + e^\top v & = 1 \\
& u, v \geq 0 & \\
& \pi \in \mathbb{R}^n & \\
& \pi_0 \in \mathbb{R} &
\end{aligned} \tag{3.128}$$

The constraint $e^\top u + e^\top v = 1$ is a normalisation constraint that ensures feasibility as the model can be unbounded. If there exits a solution (π, π_0) to model (3.128) where the objective value is positive then x^* violates the cut $\pi^\top x \leq \pi_0$ and we add this cut to P.

In the following we describe how we apply the disjunctive cuts for CTT. We apply the cuts for the basic model described in section 3. For the cuts we use the disjunction from section 3.1.1. For a fractional solution $\overline{x}$ consider curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ where $\overline{s}_{q,d,t}$ is the value of the variable $s_{q,d,t}$ in the fractional solution. We calculate the sum $\overline{x}_{q,d,t} + \overline{x}_{q,d,t-1} + \overline{x}_{q,d,t+1}$, and if the sum is zero or two then we then this is not a branching candidate. If the sum if greater than zero and less than two then we calculate the value $\overline{s}^{Disj}_{q,d,t}$ according to the disjunctive formulation:

$$\overline{s}^{Disj}_{q,d,t} = \begin{cases} \overline{x}_{q,d,t} & \text{if } \overline{x}_{q,d,t} + \overline{x}_{q,d,t-1} + \overline{x}_{q,d,t+1} \leq 1 \\ 1 - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} & \text{otherwise} \end{cases} \tag{3.129}$$

If $s^{Disj}_{q,d,t} > s_{q,d,t}$ then this is a candidate for a disjunctive cut. If the sum $\overline{x}_{q,d,t} + \overline{x}_{q,d,t-1} + \overline{x}_{q,d,t+1}$ is greater than one then we use the disjunction (3.18) for the cut separation problem (3.128). If the sum if less than one then is use the disjunction (3.25). If the sum is one then we pick one of the disjunctions randomly.

In our implementation we used Gurobi and implemented the cutting plane in a callback function. We tested the approach where we added the cuts only in the root node of the Branch & Bound tree. The strategy we used to generate the cuts were to order the candidates by the value $\overline{s}^{Disj}_{q,d,t} - \overline{s}_{q,d,t}$ from largest to smallest. We then iterated through the list and then added only the first cut that was generated. Though cuts were generated, it did not help on the performance of Gurobi. For future research it could be interesting to examine further if it helps to add multiple cuts and if the cuts should be added in other nodes in the Branch & Bound tree as well.

3.4.4 Nonlinear Disjunctive Cuts

In this section we consider disjunctive cuts similar to the cuts in section 3.4.3. However, in this section we consider a disjunction based on a nonlinear formulation of the isolated lectures.

Consider a curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot$t \in \mathcal{T}$. Curriculum q has an isolated lecture in period (d, t) if the variable $x_{q,d,t}$ is one and both $x_{q,d,t-1}$ and $x_{q,d,t+1}$ are zero. So we can replace the constraint (3.11) by the following nonlinear constraint:

$$s_{q,d,t} = x_{q,d,t}\,(1 - x_{q,d,t-1})\,(1 - x_{q,d,t+1}) \tag{3.130}$$

Consider a solution $\overline{x}$ to the LP relaxation of the basic model. We let $\overline{s}^{LP}_{q,d,t}$ be the value of the variable $s_{q,d,t}$ calculated from the linear formulation (3.11) and $\overline{s}^{NLP}_{q,d,t}$ be the value calculated from the nonlinear formulation (3.130). Unfortunately the (3.130) is nonlinear and non-convex so we cannot use it in a solver such as Gurobi. However, provided with the solution $\overline{x}$ to the LP relaxation, we can calculate the nonlinear terms. In Table 3.2 we have solved the LP relaxation for all the data instances from ITC2007. For each instance we report the objective value of the LP relaxation in the column LP and in the column NLP we report the objective value of the same solution if we had used the nonlinear formulation.

Table 3.2: For each instance the objective value of the LP relaxation (LP) is provided as well as the corresponding objective value for the non-linear formulation

Instance	LP	NLP
comp01	4.0	51.6
comp02	0.0	304.0
comp03	0.0	306.4
comp04	0.0	210.8
comp05	17.0	911.7
comp06	6.0	312.3
comp07	0.0	325.5
comp08	0.0	230.6
comp09	0.0	334.1
comp10	0.0	253.6
comp11	0.0	56.1
comp12	3.0	979.0
comp13	0.0	262.4
comp14	0.0	269.4
comp15	0.0	306.4
comp16	4.0	319.1
comp17	10.0	331.6
comp18	0.0	284.6
comp19	4.5	271.8
comp20	0.0	363.3
comp21	0.0	364.8

From Table (3.2) we see that there is a large difference in the objective values, which is always the case. If we expand the expression (3.130) for the solution $\overline{x}$ we get the following:

$$\overline{x}_{q,d,t}\,(1 - \overline{x}_{q,d,t-1})\,(1 - \overline{x}_{q,d,t+1}) = \overline{x}_{q,d,t} - \overline{x}_{q,d,t}\overline{x}_{q,d,t-1} - \overline{x}_{q,d,t}\overline{x}_{q,d,t+1} + \overline{x}_{q,d,t}\overline{x}_{q,d,t-1}\overline{x}_{q,d,t+1} \tag{3.131}$$

As all three variables are non-negative then $\overline{x}_{q,d,t}\overline{x}_{q,d,t-1}\overline{x}_{q,d,t+1} \geq 0$ must hold. Furthermore, since $0 \leq \overline{x}_{q,d,t} \leq 1$ then $\overline{x}_{q,d,t}\overline{x}_{q,d,t-1} \leq \overline{x}_{q,d,t-1}$ and $\overline{x}_{q,d,t}\overline{x}_{q,d,t+1} \leq \overline{x}_{q,d,t+1}$ must hold as well, and so we can calculate a lower bound for the right-hand side of (3.131):

$$\overline{x}_{q,d,t} - \overline{x}_{q,d,t}\overline{x}_{q,d,t-1} - \overline{x}_{q,d,t}\overline{x}_{q,d,t+1} + \overline{x}_{q,d,t}\overline{x}_{q,d,t-1}\overline{x}_{q,d,t+1} \geq \overline{x}_{q,d,t} - \overline{x}_{q,d,t-1} - \overline{x}_{q,d,t+1} \tag{3.132}$$

Note that the right-hand side of (3.132) is equivalent to the linear formulation of the isolated lectures. So for any optimal solution $\overline{x}$ to the LP relaxation of the basic model we have that $\overline{s}^{NLP}_{q,d,t} \geq \overline{s}^{LP}_{q,d,t}$ for every curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$. We use this information to create the disjunction and to find the candidates for the disjunction. If we have selected a curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ as the candidate then we create the disjunction by ensuring that in each subproblem the value $\overline{s}^{NLP}_{q,d,t}$ is equal to $\overline{s}^{LP}_{q,d,t}$ for the optimal solution of the LP relaxation of the subproblem.

The question is when $\overline{s}^{NLP}_{q,d,t} = \overline{s}^{LP}_{q,d,t}$ holds. If we fix $x_{q,d,t-1}$ to one then $\overline{s}^{NLP}_{q,d,t}$ is zero for any solution $\overline{x}$ and the right-hand side in (3.11) is less than or equal to zero meaning that $\overline{s}^{LP}_{q,d,t}$ is zero as well. So we can create one subproblem where we add the constraint $x_{q,d,t} \geq 1$. Then we also have to create another subproblem where we add the constraint $x_{q,d,t} \leq 0$, which creates a disjunction of the problem. In this latter suproblem we cannot guarantee that $\overline{s}^{NLP}_{q,d,t} = \overline{s}^{LP}_{q,d,t}$ for any solution. So we cretae a new disjunction of that subproblem that split it into two new subproblems. In the first subproblem we add the constraint $x_{q,d,t+1} \geq 1$ and as before $\overline{s}^{NLP}_{q,d,t} = \overline{s}^{LP}_{q,d,t} = 0$ in this subproblem. In the second subproblem we add the constraint $x_{q,d,t+1} \leq 0$. So now we have a subproblem where we have the constraints $x_{q,d,t-1} \leq 0$ and $x_{q,d,t+1} \leq 0$ which means that for a solution $\overline{x}$ the right-hand side in (3.11) evaluates to $\overline{x}_{q,d,t}$. The value $\overline{s}^{NLP}_{q,d,t}$ also evaluates to $\overline{x}_{q,d,t}$ which means that $\overline{s}^{LP}_{q,d,t} = \overline{s}^{NLP}_{q,d,t}$ for any solution $\overline{x}_{q,d,t}$. We provide an overview of the disjunction in Figure 3.7 where P refers to the problem we consider and P^L, P^M and P^R refers to the subproblems created. The label below each node is the constraint that is added to the subproblem.

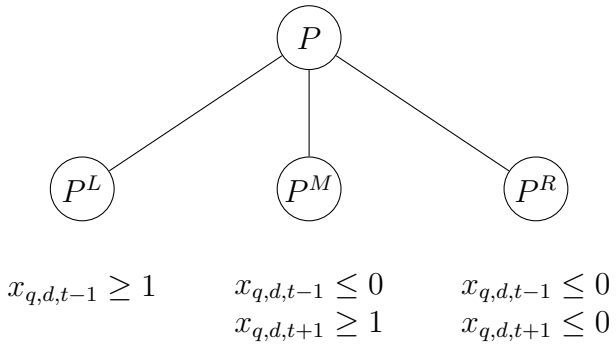


Figure 3.7: Illustration of the three children when branching

We let $A^L \leq b^L$ be the constraint set of the model in the first child P^L, i.e., if $Ax \leq b$ is the constraint set of P then $A^L \leq b^L$ is the same constrains set plus the additional constraint $x_{q,d,t-1} \geq 1$. $A^M \leq b^M$ is the constraint set of the model in the second child P^M, and $A^R \leq b^R$ is the constraint set of the model in the third child P^R. We then extend the cut model from section (3.4.3) to the following:

$$
\begin{aligned}
\max \quad & \pi^\top x^* - \pi_0 & \\
\text{s.t.} \quad & \pi & \leq uA^L \\
& \pi & \leq vA^M \\
& \pi & \leq wA^R \\
& \pi_0 & \geq ub^L \\
& \pi_0 & \geq vb^M \\
& \pi_0 & \geq wb^R \\
& e^\top u + e^\top v + e^\top w & = 1 \\
& u, v, w \geq 0 & \\
& \pi \in \mathbb{R}^n & \\
& \pi_0 \in \mathbb{R} &
\end{aligned}
\tag{3.133}
$$

We implemented the cutting plane as a callback function in Gurobi. Whenever we were provided with a fractional solution $\overline{x}$ by Gurobi in the root node of the Branch & Bound tree, we ordered each triple (q, d, t) for curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ for the value $\overline{s}^{NLP}_{q,d,t} - \overline{s}^{LP}_{q,d,t}$ from largest to smallest. We then iteration two the triples and for each triple we solved the cut separation model (3.133), and if a cut is generated, we add it to the model and stop. The drawback of this method compared to the disjunctive cuts in section 3.4.3 is that the cut separation model is larger, which made it more time consuming to solve. It could be interesting to see if it help the performance if all the violated cuts and also if the solution time of the cut separation process can be improved.

3.5 Dantzig-Wolfe Decomposition

In this section we describe different Dantzig-Wolfe Decomposition that we have considered. Before the description of our implementations, we provide an introduction to Dantzig-Wolfe Decomposition in section 3.5.1. This introduction is taken directly from the paper in chapter 7.

Dantzig-Wolfe decompositions have been applied to CTT before, e.g., Cacchiani et al. (2013) mentions different formulations. In one of the decompositions, they decompose one of the subproblems further when the data instances are large such that there is a subproblem for each day. In the paper in chapter 7 a decomposition is described that also generates a subproblem for each day. The difference between Cacchiani et al. (2013) and the paper in chapter 7 is that the approach in chapter 7 based on a pattern formulation described in the paper in chapter 6.

In section 3.5.2 and 3.5.3 we describe two other decompositions that we have implemented which are also based on the pattern formulation from the paper in chapter 6. The first approach is to decompose the model according to the courses and the second approach divides the course into cliques and then decomposes the model according to those cliques.

3.5.1 Introduction to Dantzig-Wolfe Decomposition

Our introduction to the Dantzig-Wolfe Decomposition is specific for MIP models and we refer to Martin (1999, chapter 11) and Desrosiers and Lübbecke (2010) for thorough and more general

descriptions. We consider an MIP of the form:

$$\begin{aligned} \min \quad & c^\top x && (3.134)\\ \text{s.t.} \quad & Ax \geq b && (3.135)\\ & Bx \geq d && (3.136)\\ & x \in \mathbb{Z}^n && (3.137) \end{aligned}$$

where c, x, b and d are vectors and A and B are matrices. To get a lower bound of the model (3.134) – (3.137) the linear programming (LP) relaxation is solved. In Figure 3.8 an example of the solution space is illustrated.

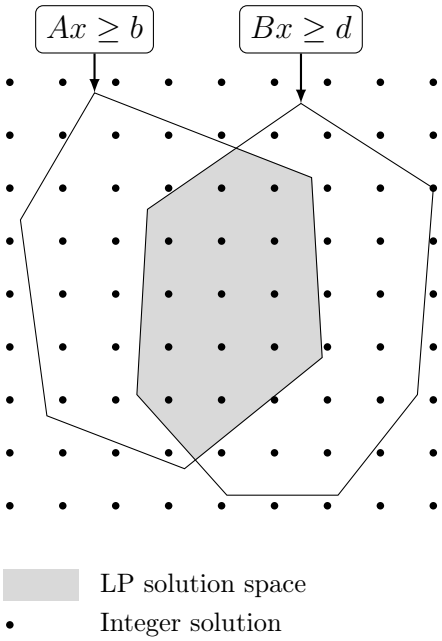


Figure 3.8: Illustration of the solution space.

The idea of the Dantzig-Wolfe decomposition for an MILP is to take the convex hull of $\{x \in X \mid Bx \geq d\}$, and replace it by variables. For simplicity, we assume that the convex hull $conv\,(\{x \in X \mid Bx \geq d\})$ is a polytope. Then any point $\overline{x}$ in this polytope can be written as a convex combination of the extreme points, i.e., if $\left\{\overline{x}^h\right\}_{h\in\mathcal{H}}$ is the set of all the extreme points of $conv\,(\{x \in \mathbb{R} \mid Bx \geq d\})$, then $\overline{x}$ can be written as follows:

$$\begin{aligned} & \sum_{h\in\mathcal{H}} \overline{x}^h \lambda^h = \overline{x} && (3.138)\\ & \sum_{h\in\mathcal{H}} \lambda^h = 1 && (3.139)\\ & \lambda^h \geq 0, \quad \forall h \in \mathcal{H} && (3.140) \end{aligned}$$

We can take this representation and insert it into the LP-relaxation of the model (3.134) – (3.137):

$$\min \quad \sum_{h \in \mathcal{H}} \left(c^\top \overline{x}^h\right) \lambda^h \tag{3.141}$$

$$\text{s.t.} \quad \sum_{h \in \mathcal{H}} \left(A\overline{x}^h\right) \lambda^h \geq b \tag{3.142}$$

$$\sum_{h \in \mathcal{H}} \lambda^h = 1 \tag{3.143}$$

$$\lambda^h \geq 0, \quad \forall h \in \mathcal{H} \tag{3.144}$$

Model (3.141) – (3.144) is referred to as the LP relaxation of the Dantzig-Wolfe *master problem*. Let z^*_{IP} be the optimal objective value of the model (3.134) – (3.137), let z^*_{LP} be the optimal objective value of the LP relaxation of the same model and let z^*_{DW} be the optimal objective value of the model (3.141) – (3.144). Then the benefit of rewriting the model is that the LP relaxation of the Dantzig-Wolfe master problem is a stronger relaxation in the sense that we have the following relation:

$$z^*_{LP} \leq z^*_{DW} \leq z^*_{IP} \tag{3.145}$$

Figure 3.9 illustrates the impact on the solution space when $Bx \geq d$ from Figure 3.8 is replaced by $conv\left(\{Bx \geq d, x \in \mathcal{Z}^n\}\right)$.

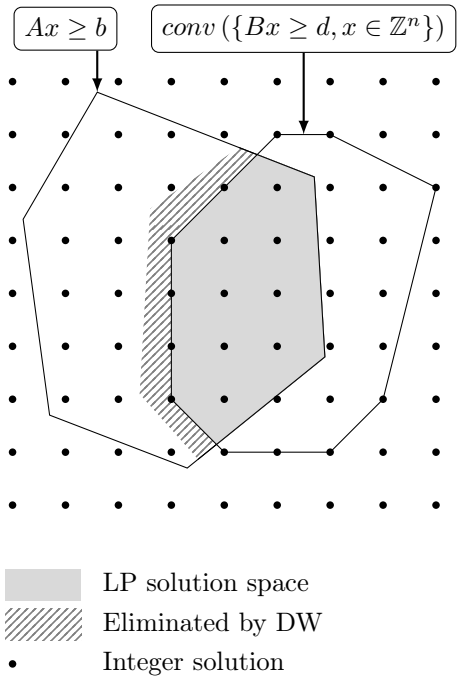


Figure 3.9: Illustration of the solution space of the master problem.

Explicitly describing model (3.141) – (3.144) can be difficult as the number of extreme points in $conv\left(\{x \in X \mid Bx \geq d\}\right)$ can be exponentially large. So a way to solve the model is to start with a restricted set $\mathcal{H}' \subseteq \mathcal{H}$ and then solve the model (3.141) – (3.144) with this restricted set.

The model with this restricted set is referred to as the *restricted master problem* (RMP). Let π be the dual vector of the constraints (3.142) and π_0 be the dual variable of constraint (3.143). For a dual solution $(\overline{\pi}, \overline{\pi}_0)$ the reduced cost of a column $h \in \mathcal{H}$ is $\overline{c}^h = (c - A^\top \overline{\pi})^\top \overline{x}^h - \overline{\pi}_0$. If the dual solution is optimal then the RMP is optimal when $\overline{c}^h \geq 0$ for every $h \in \mathcal{H}$. We know that $\overline{c}^h \geq 0$ for every $h \in \mathcal{H}'$, but there may exist some column $h \in \mathcal{H} \backslash \mathcal{H}'$ where $\overline{c}^h < 0$ so we need to check if any such column exist. This can be done by solving the following problem, known as the *pricing problem* (PP):

$$\begin{aligned} \min \quad & (c - A^\top \overline{\pi})^\top x - \overline{\pi}_0 && (3.146) \\ \text{s.t.} \quad & Bx \geq d && (3.147) \\ & x \in \mathbb{Z}^n && (3.148) \end{aligned}$$

If PP contains any solution where the objective value is negative, then the RMP is not proven optimal, and we need to add the solution as a column to the RMP. This process is known as the *column generation* algorithm. First solve the RMP to obtain the dual solution $(\overline{\pi}, \overline{\pi}_0)$. Given the dual solution find a solution for model (3.146) – (3.148) with a negative objective value. If a solution with a negative reduced cost exists, then extend the restricted set $\mathcal{H}'$ with this solution and iterate the process. We continue this iterative process until the model (3.146) – (3.148) does not contain any solution with a negative objective value. This iterative process is illustrated in Figure 3.10.

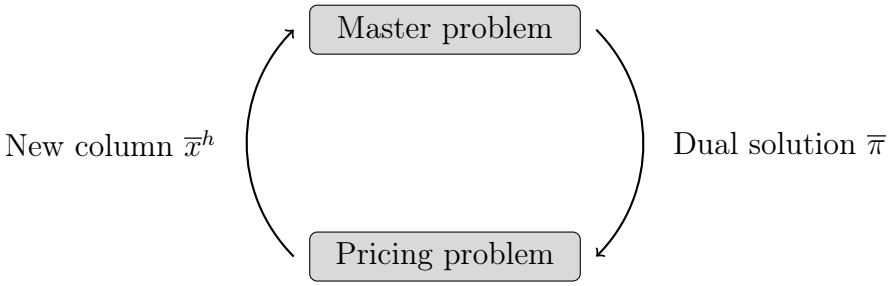


Figure 3.10: The iterative loop of the column generation algorithm.

Another benefit of the Dantzig-Wolfe decomposition is that it is possible to exploit if the constraint matrix B has a *block diagonal* structure as follows:

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_m \end{bmatrix}, \quad d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} \tag{3.149}$$

Let $\mathcal{I} = \{1, 2, \ldots, m\}$ and for each $i \in \mathcal{I}$ let $\mathcal{H}_i$ be the extreme points of $conv\left(\{x_i \in \mathbb{Z} \,|\, B_i x_i \geq d_i\}\right)$ where x_i is the subset of variables in x that corresponds to the submatrix B_i. Similarly, c_i and A_i are the subvector and submatrix of the vector c and matrix A corresponding to the submatrix B_i. Then the LP relaxation of the master problem can be written as follows:

$$\min \quad \sum_{i\in\mathcal{I},h\in\mathcal{H}_i} \left(c_i^\top \overline{x}_i^h\right) \lambda_i^h \tag{3.150}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I},h\in\mathcal{H}_i} \left(A_i \overline{x}_i^h\right) \lambda_i^h \geq b \tag{3.151}$$

$$\sum_{h\in\mathcal{H}_i} \lambda_i^h = 1, \quad \forall i \in \mathcal{I} \tag{3.152}$$

$$\lambda_i^h \geq 0, \quad \forall i \in \mathcal{I}, h \in \mathcal{H}_i \tag{3.153}$$

The columns with a negative reduced cost are then found by solving the m independent pricing problems.

3.5.2 Course Schedules

The first decomposition we considered for the pattern formulation from the paper in chapter 6 was to create a pricing problem for each day. The befit of this decomposition is that the pricing problem can be solved efficiently by *Dynamic Programming*. Dynamic Programming is a method for solving a problem by solving it in *stages*. Each stage is given the *state* of the previous stages to calculate new states to be provided for succeeding stages. In our implementation a state consists if three values; $\overline{c}$, L and M. $\overline{c}$ is the cost of the state. L corresponds to the number of all the lectures that has been scheduled for the course in previous stages. M is the number of days left for the course to schedule lectures to hit the target of the minimum working day $D_c^{\min}$. In the beginning the state is $\left(\overline{c}, M = D_c^{\min}, L = 0\right)$ where $\overline{c}$ is set to be equal to any constant of the objective function for the pricing problem. The stages correspond to the days. When a stage (day) gets a state $(\overline{c}_i, M_i, L_i)$ as input, then for each pattern that the course can be assigned on that day, a new state $(\overline{c}_j, M_j, L_j)$ is created. $\overline{c}_j$ is set to be equal to $\overline{c}_i$ plus the cost of the pattern. L_j is set to be equal to L_i plus the number of lectures in the pattern. $M_j \leftarrow M_i$ if the pattern does not contain any lectures, otherwise $M_j \leftarrow (M_i - 1)^+$. For two different states $(\overline{c}_{j_1}, M_{j_1}, L_{j_1})$ and $(\overline{c}_{j_2}, M_{j_2}, L_{j_2})$ produced by the same stage, we say that $(\overline{c}_{j_1}, M_{j_1}, L_{j_1})$ *dominates* $(\overline{c}_{j_2}, M_{j_2}, L_{j_2})$ if $L_{j_1} = L_{j_2}$ and $\overline{c}_{j_1} + W^{\mathbf{MWD}} (M_{j_1} - M_{j_2})^+ \leq \overline{c}_{j_2}$. The dominated label can be discarded as it cannot lead to a solution which is better than the best solution that is created from $(\overline{c}_{j_1}, M_{j_1}, L_{j_1})$. Any state $(\overline{c}, M, L)$ returned by the final stage is feasible if $L = L_c$. For each feasible state $(\overline{c}, M, L)$ we add the cost of the minimum working days; $\overline{c} \leftarrow \overline{c} + W^{\mathbf{MWD}} M$. We can then pick the feasible state with the lowest cost as the optimal solution. An example of the process is illustrated in Figure 3.11, where Monday is provided with the initial state. All the states generated by Monday is then provided as input for Tuesday. The algorithm continues until Friday has generated all its states.

In the example in Figure 3.11 we processed the stages starting from Monday and ended on Friday. However, we could have processed the stages in any order we like. Therefore, we implemented a faster dynamic algorithm by generating the states for each day independently and then recursively *merging* the states together. Consider two states $(\overline{c}_i, M_i, L_i)$ and $(\overline{c}_j, M_j, L_j)$ that were generated by two different days. We can merge the states into stage $(\overline{c}_k, M_k, L_k)$ if, and only if, $L_i + L_j \leq L_c$, otherwise $(\overline{c}_k, M_k, L_k)$ becomes infeasible. We set $L_k = L_i + L_j$. For the values $\overline{c}_k$ and M_k we need to consider the number of days that the course has lectures assigned

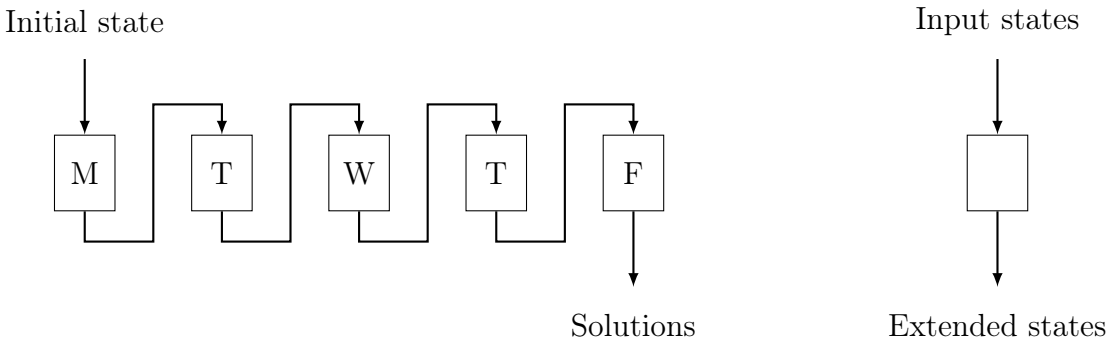


Figure 3.11: Illustration of the dynamic programming algorithm

in the two states. For the state $(\overline{c}_i, M_i, L_i)$ the number of working days is $D_c^{\min} - M_i$, and for the state $(\overline{c}_j, M_j, L_j)$ the number is $D_c^{\min} - M_j$. So when we merge the labels the total number of working days is $D_c^{\min} - M_i + D_c^{\min} - M_j = 2D^{\min} - M_i - M_j$. Now we can calculate the number of days missing for the course, $M_k = \left(D^{\min} - 2D^{\min} + M_i + M_j\right)^+ = \left(M_i + M_j - D_c^{\min}\right)^+$. An example of the complete merge procedure is illustrated in Figure 3.12.

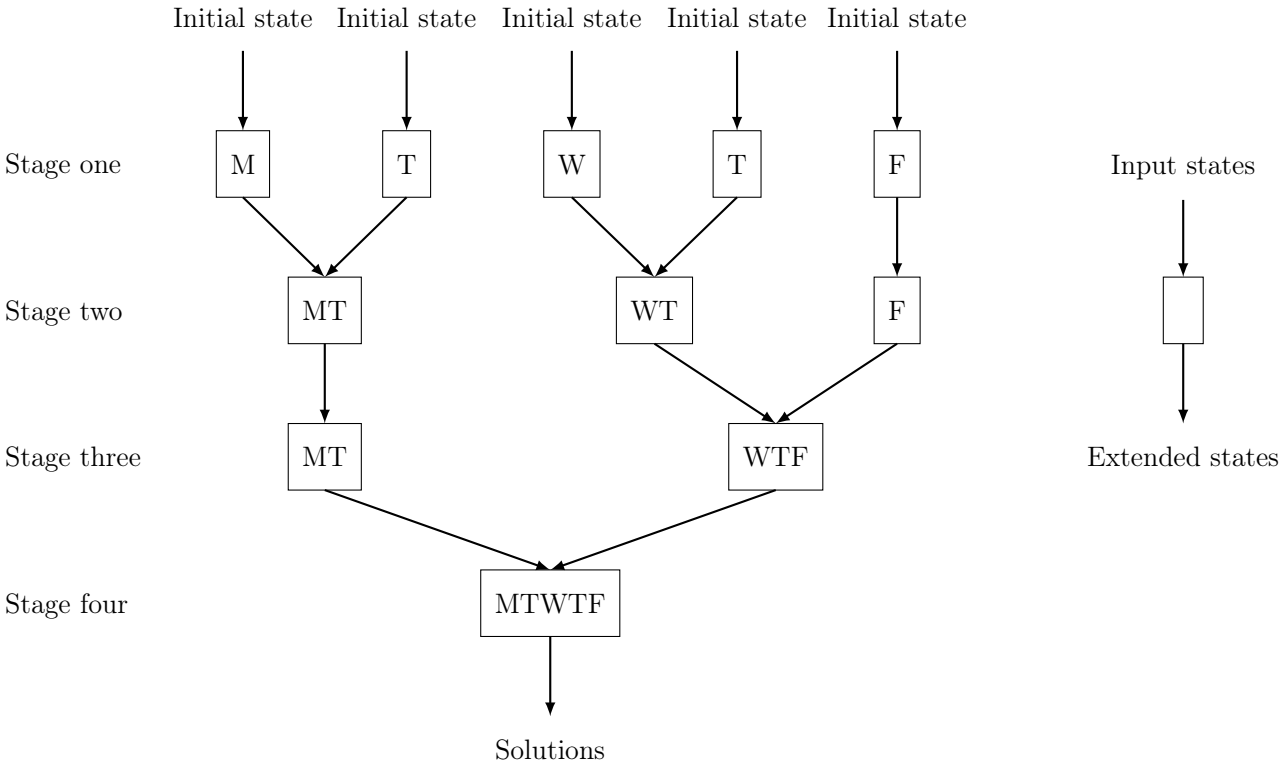


Figure 3.12: Illustration of the state merging procedure

In Figure 3.12 we start the first stage by providing each day with the initial state. Each day then generates a list of states. Then for each state in the list returned by Monday, we merge it with each state in the list returned by Tuesday, which generates a new list (denoted MT

in the figure) where the dominated states are discarded. We do the same for Wednesday and Thursday (denoted WT in the figure). At this stage we carry over the list of states generated by Friday (denoted F in the figure), i.e., the states generated by Friday are not merged. In the second stage we carry over the list MT to the next stage, and we merge the lists WF and F into the list WTF. In the third stage we merge the two lists, MT and WTF, which generates the list MTWTF. In the last stage we remove all the states where the number of lectures is not L_c, and we can then pick the state with the lowest cost as our solution.

3.5.3 Clique Schedules

In this section we describe a decomposition based on the cliques Γ. The formulation is very similar to the course decomposition in section 3.5.2, where we also consider the pattern formulation from the paper in chapter 6. However, instead of decomposing the model, so we have a pricing problem for each course, we decompose the problem such that each pricing problem corresponds to a clique of courses. The benefit of this decomposition is that for a clique of courses we know that they must be scheduled in different periods. Since each pricing problem corresponds to a clique, we restrict the columns that are generated by only selecting patterns for the courses that schedule at most one lecture in each period. The idea is to decompose the courses into cliques such that each course is contained in exactly one of the cliques. We take the cliques found by the algorithm described by Bron and Kerbosch (1973) and define a binary variable x_γ for each clique $\gamma \in \Gamma$ which is one if the clique is selected for the decomposition, and zero otherwise. For each clique $\gamma \in \Gamma$ we define the value α_γ, which is some measurement of how profitable it is to include γ in the decomposition. We then formulate the decomposition problem as follows:

$$\begin{aligned} \max \quad & \sum_{\gamma \in \Gamma} \alpha_\gamma x_\gamma \\ \text{s.t.} \quad & \sum_{\gamma \in \Gamma_c} x_\gamma = 1 \quad \forall c \in \mathcal{C} \\ & x \in \mathbb{B} \end{aligned} \tag{3.154}$$

In the model (3.154) there is one constraint for each course, which ensures that the course is contained in at least one of the selected cliques. For the optimal solution to model (3.154) we decompose the pattern formulation

To ensure that the model (3.154) is feasible we add a clique to Γ for each course $c \in \mathcal{C}$, which contains only this single course. Let $\mathcal{C}_{q,d,t}$ be defined for curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ as the set of courses that can be scheduled in one of the time slots $\{t-1, t, t+1\}$:

$$\mathcal{C}_{q,d,t} := \{c \in \mathcal{C}_q \mid F_{c,d,t-1} + F_{c,d,t} + F_{c,d,t+1} \geq 1\}, \quad q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{3.155}$$

If all the courses in the set $\mathcal{C}_{q,d,t}$ is in a clique $\gamma \in \Gamma$, then we can put the formulation of the isolated lecture in the corresponding pricing problem. Therefore, we define the profit of selecting clique $\gamma \in \Gamma$ to be the number of sets (3.155) where all courses are in the clique:

$$\alpha_\gamma := \|\{q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \,:\, \mathcal{C}_{q,d,t} \subseteq \mathcal{C}_\gamma\}\| \tag{3.156}$$

For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ we can also add the set of courses $\mathcal{C}_{q,d,t}$ to the set of cliques since $\mathcal{C}_{q,d,t} \subseteq \mathcal{C}_q$. We add these cliques to increase the chances of the isolated lectures to be included in the pricing problems.

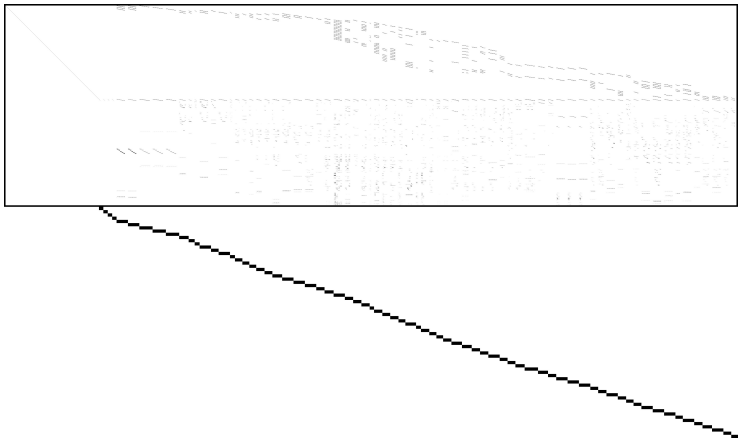
3.5.4 Remarks on the Decompositions

The two formulations from section 3.5.2 and 3.5.3 were not as successful as the decomposition from the paper in chapter 6. To get an idea of why this could be the case, we consider the constraint matrix of the pattern formulation. In Figure 3.13 we illustrate the constraint matrix of the pattern formulation for the data instance comp03 from ITC2007. Each black pixel, except the rectangles, corresponds to a non-zero entry in the matrix. In Figure 3.12a we have ordered the columns and rows according to courses, in Figure 3.12b we have ordered the columns and rows according to the clique decomposition and in Figure 3.12c the columns an rows are ordered by days. The rectangles mark the constraints for the master problem and the block structure of the pricing problem.

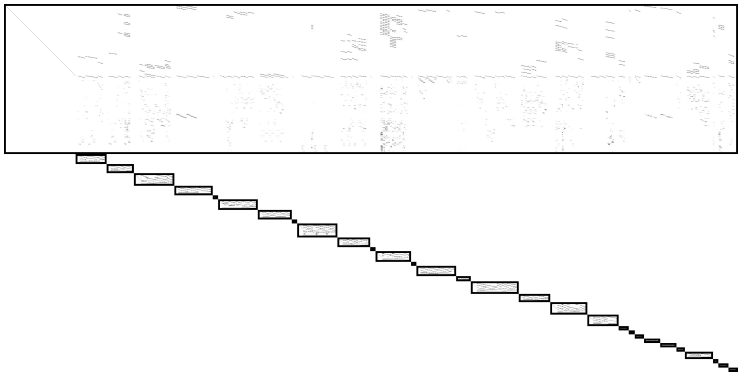
It should be noted that for the sake of visualization we added a border around each block in Figure 3.13. The width of the border is four which is added to the sizes of the matrices. Therefore, the total number of constraints in the pricing problems may appear to be larger than they are. For instance in Figure 3.12a the total amount of constraints in the pricing problems appear to be more than half of all the constraints, though, in reality only 5% of the constraints are in the pricing problems. For the clique decomposition, 43% of the constraints are in the pricing problems and for the day decomposition, 89% of the constraints are in the pricing problem.

It is mentioned by Martin (1999, chapter 11) that to take full advantage of the Dantzig-Wolfe Decomposition procedure the pricing problem must contain a *vast majority* of the constraints, which is the case for the day decomposition, but not for the other decompositions. Another reason that the day decomposition is more successful than the other two is due to the isolated lectures. In the course decomposition all of the isolated lectures are formulated in the master problem, and for the clique decomposition, a few of them are contained in the pricing problems. For the day decomposition, the isolated lectures are contained entirely in the pricing problems.

(a) Course decomposition



(b) Clique decomposition



(c) Day decomposition

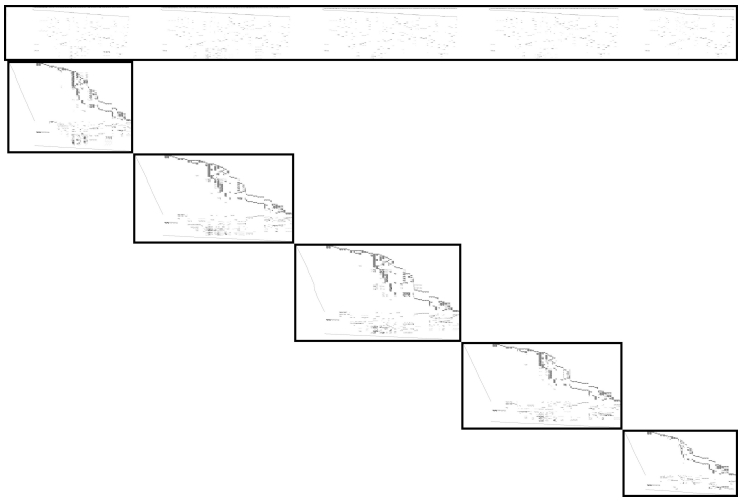


Figure 3.13: Illustration of the block diagonal structures for data instance comp03 when ordering by courses (a), cliques (b) or days (c). Figure (c) is taken from the paper in chapter 7

References

Asín Aschá, R. and Nieuwenhuis, R. (2014). "Curriculum-based course timetabling with SAT and MaxSAT". In: *Annals of Operations Research* 218, pp. 71–91.

Bærentsen, R. (2012). "Optimization of room-allocation at the Technical University of Denmark". PhD thesis. Technical University of Denmark.

Bagger, N., Kristiansen, S., Sørensen, M., and Stidsen, T. (2015). "Flow Formulation-based Model for the Curriculum-based Course Timetabling Problem". In: *MISTA 2015 Proceedings*. Ed. by Z. Hanzálek, G. Kendall, B. McCollum, and P. Šøucha. Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2015), pp. 825–848.

Benders, J. (1962). "Partitioning procedures for solving mixed-variables programming problems". English. In: *Numerische Mathematik* 4.1, pp. 238–252. ISSN: 0029-599X. DOI: 10.1007/BF01386316. URL: http://dx.doi.org/10.1007/BF01386316.

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). "An overview of curriculum-based course timetabling". In: *TOP* 23.2, pp. 313–349.

Bonutti, A., De Cesco, F., Di Gaspero, L., and Schaerf, A. (2012). "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results". In: *Annals of Operations Research* 194.1, pp. 59–70.

Bron, C. and Kerbosch, J. (1973). "Algorithm 457: Finding All Cliques of an Undirected Graph". In: *Communications of the ACM* 16.9, pp. 575–577. DOI: 10.1145/362342.362367. URL: http://doi.acm.org/10.1145/362342.362367.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2008). "Penalising Patterns in Timetables: Novel Integer Programming Formulations". In: *Operations Research Proceedings 2007*. Springer, pp. 409–414.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010a). "A supernodal formulation of vertex colouring with application in course timetabling". English. In: *Annals of Operations Research* 179.1, pp. 105–130. ISSN: 0254-5330.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010b). "Decomposition, reformulation, and diving in university course timetabling". In: *Computers & Operations Research* 37.3, pp. 582–597. DOI: 10.1016/j.cor.2009.02.023.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). "A branch-and-cut procedure for the Udine Course Timetabling problem". In: *Annals of Operations Research* 194.1, pp. 71–87.

Cacchiani, V., Caprara, A., Roberti, R., and Toth, P. (2013). "A new lower bound for curriculum-based course timetabling". In: *Computers and Operation Research* 40.10, pp. 2466–2477. DOI: 10.1016/j.cor.2013.02.010.

Desrosiers, J. and Lübbecke, M. E. (2010). "A Primer in Column Generation". In: *Column Generation*. Ed. by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Springer Science+Business Media, Inc. Chap. 1, pp. 1–32. ISBN: 978-1-4419-3799-5.

Di Gaspero, L., Schaerf, A., and McCollum, B. (2007). "The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3) — preliminary presentation —". In: *Association for the Advancement of Artificial Intelligence (www.aaai.org)*.

Fischetti, M., Lodi, A., and Tramontani, A. (2011). "On the separation of disjunctive cuts". In: *Mathematical Programming* 128.1, pp. 205–230. ISSN: 1436-4646. DOI: 10.1007/s10107-009-0300-y. URL: http://dx.doi.org/10.1007/s10107-009-0300-y.

Gamrath, G., Fischer, T., Gally, T., Gleixner, A. M., Hendel, G., Koch, T., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Vigerske, S., Weninger, D., Winkler, M., Witt, J. T., and Witzig, J. (2016). *The SCIP Optimization Suite 3.2*. eng. Tech. rep. 15-60. Takustr.7, 14195 Berlin: ZIB.

Geoffrion, A. M. (1972). "Generalized benders decomposition". In: *Journal of optimization theory and applications* 10.4, pp. 237–260.

Gurobi Optimization Inc. (2015). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Gurobi Optimization, Inc. (2016). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Hao, J. K. and Benlic, U. (2011). "Lower bounds for the ITC-2007 curriculum-based course timetabling problem". In: *European Journal of Operational Research* 212.3, pp. 464–472.

International Business Machines Corp. (2017). *IBM ILOG CPLEX Optimization Studio V12.7.0 documentation*.

Lach, G. and Lübbecke, M. E. (2008). "Optimal University Course Timetables and the Partial Transversal Polytope". In: *International Workshop on Experimental and Efficient Algorithms*. Springer, pp. 235–248.

Lach, G. and Lübbecke, M. E. (2012). "Curriculum based course timetabling: New solutions to Udine benchmark instances". In: *Annals of Operations Research* 194.1, pp. 255–272.

Lewis, R., Paechter, B., and McCollum, B. (2007). *Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition*. Cardiff Accounting and Finance Working Papers A2007/3. Cardiff University, Cardiff Business School, Accounting and Finance Section.

Martin, R. K. (1999). *Large scale linear and integer optimization: a unified approach*. Kluwer Academic Publishers.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Di Gaspero, L., Qu, R., and Burke, E. K. (2010). "Setting the research agenda in automated timetabling: The second international timetabling competition". In: *INFORMS Journal on Computing* 22.1, pp. 120–130.

Padberg, M. W. (1974). "Perfect zero–one matrices". In: *Mathematical Programming* 6.1, pp. 180–196.

Phillips, A. E., Waterer, H., Ehrgott, M., and Ryan, D. M. (2015). "Integer programming methods for large-scale practical classroom assignment problems". In: *Computers & Operations Research* 53, pp. 42–53. ISSN: 03050548. DOI: 10.1016/j.cor.2014.07.012.

Pillay, N. (2016). "A review of hyper-heuristics for educational timetabling". In: *Annals of Operations Research* 239.1, pp. 3–38. ISSN: 1572-9338. DOI: 10.1007/s10479-014-1688-1. URL: http://dx.doi.org/10.1007/s10479-014-1688-1.

Ryan, D. M. and Falkner, J. C. (1988). "On the integer properties of scheduling set partitioning models". In: *European journal of operational research* 35.3, pp. 442–456.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). "Towards Objective Measures of Algorithm Performance Across Instance Space". In: *Comput. Oper. Res.* 45, pp. 12–24. ISSN: 0305-0548. DOI: 10.1016/j.cor.2013.11.015. URL: http://dx.doi.org/10.1016/j.cor.2013.11.015.

The Scheduling and Timetabling Research Group at the University of Udine, Italy (2015). *Curriculum-Based Course TimeTabling*. Last retrieved October 2015, http://tabu.diegm.uniud.it/ctt/index.php. URL: http://tabu.diegm.uniud.it/ctt/index.php.

Wolsey, L. A. (1998). "Cutting Plane Algorithms". In: *Integer Programming*. Ed. by R. Graham, J. Lenstra, and R. Tarjan. John Wiley & Sons, Inc. Chap. 8, pp. 113–137. ISBN: 0-471-28366-5.

Part II

Exact Methods

4 Flow Formulations for Curriculum-based Course Timetabling

Niels-Christian F. Bagger[a,b] · Simon Kristiansen[c] · Matias Sørensen[a,b] · Thomas R. Stidsen[a]

[a]mORetime research group, Management Science, Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 426B, DK-2800 Kgs. Lyngby, Denmark, http://www.moretime.man.dtu.dk/

[b]MaCom A/S, Vesterbrogade 48, 1., DK-1620 København V, Denmark

[c]RHA Software Group, Frederikkevej 2B, DK-2900 Hellerup Denmark

Status: Submitted to Annals of Operations Research

Abstract: In this paper we present two mixed-integer programming formulations for the Curriculum based Course Timetabling Problem (CTT). We show that the formulations contain underlying network structures by dividing the CTT into two separate models and then connect the two models using flow formulation techniques. The first mixed-integer programming formulation is based on an underlying minimum cost flow problem, which decreases the number of integer variables significantly and improves the performance compared to an intuitive mixed-integer programming formulation. The second formulation is based on a multi-commodity flow problem which in general is NP-hard, however, we prove that it suffices to solve the linear programming relaxation of the model. The formulations show competitiveness with other approaches based on mixed-integer programming from the literature and improve the currently best known lower bound on one data instance in the benchmark data set from the second international timetabling competition. Regarding upper bounds, the formulation based on the minimum cost flow problem performs better on average than other mixed integer programming approaches for the CTT.

Keywords: University Course Timetabling · Integer Programming · Minimum Cost Flow · Multi-Commodity Flow

4.1 Description and Literature

Each semester, universities face the problem of generating high-quality course timetables. A timetable determines when and where a course should take place. In this work we consider the

Curriculum based Course Timetabling (CCT) Problem in which weekly lectures for multiple courses have to be scheduled and assigned to rooms. A week is divided into days (usually five or six) and each day is divided into time slots. We refer to a day and time slot combination as a period. The problem was introduced in track 3 of the second international timetabling competition (ITC2007) as described by Di Gaspero et al. (2007), McCollum et al. (2010) and Bonutti et al. (2012).

The basic entities of the problem are the *courses* to schedule, and the *periods* and *rooms* that are available. Therefore, when formulating a Mixed Integer Program (MIP), it seems natural to define binary variables with three indices corresponding to a course, period and room where the binary variable would then take value one if the course is scheduled in the specified room at the period. Formulating the MIP of the problem this way is the most commonly used, see Lach and Lübbecke (2008). We show that such a formulation will create an unnecessarily large amount of binary variables and instead we formulate two MIPs containing a significantly smaller number of integer variables. The goal of the formulations is to decrease the number of integer variables, and we show that we can define some of the variables as continuous variables instead of integer variables.

Besides the courses, periods and rooms the problem also contains *lecturers* and *curricula*, hence the name *Curriculum-based Course Timetabling*. Each course is taught by a lecturer, and a curriculum is a set of courses which may be followed by the same students. The resulting timetable must fulfil some specific hard constraints:

Availability (**A**): For each course a subset of the periods (maybe all the periods) are denoted as the *available* periods. It is not allowed to schedule a lecture in a period that is not available for the corresponding course.

Lectures (**L**): All lectures of all courses must be scheduled, and lectures must be scheduled in different periods.

Conflicts (**C**): If two courses are taught by the same lecturer or the courses are part of the same curriculum, the courses cannot be taught in the same period.

Room Occupancy (**RO**): A room can at most be occupied by one course in any period.

Besides the hard constraints, the problem also contains the following soft constraints:

Room Capacity (**RC**): If a lecture is scheduled in a room where the capacity is smaller than the number of students attending the course, then each student above the capacity is counted as a violation.

Isolated Lectures (**IL**): It is desired to schedule lectures from the same curriculum in adjacent periods. Two periods are considered to be adjacent if they belong to the same day and are in consecutive time slots. If a lecture from a curriculum is scheduled in a period and no lecture from the same curriculum is scheduled in an adjacent period, the lecture is denoted as being *isolated*. Every time there is an isolated lecture this counts as one violation.

Minimum Working Days (**MWD**): For each course, it is desired to spread the lectures across a given number of days. If the number of days that a course is scheduled is below this

number, then the violation is the difference between the requested and the actual number of days that has been scheduled.

Room Stability (**RStab**): Each course should not be assigned to too many different rooms during the week. If a course is scheduled in at least two distinct rooms during the week, then the violation is the total number of distinct rooms assigned to the course minus one.

The objective of the CTT is to find a solution which fulfils all the hard constraints and minimizes a weighted sum of the violations of the soft constraints using non-negative weights.

Many researchers have considered the CTT since the ITC2007 competition, and we refer to Bettinelli et al. (2015) for an excellent overview. Since we are formulating a MIP model of the problem, we mainly focus on other MIP based approaches in the literature in the remainder of this paper.

Burke et al. (2008) and Burke et al. (2010) introduced a MIP formulation named the *monolithic* formulation, based on the intuition of the three-index binary variables (for each course, period and room a binary variable is defined). The monolithic formulation is an exact model in the sense that it can be solved by a generic MIP solver to obtain the optimal solution, assuming that enough computational resources are available for the MIP solver. Unfortunately, many instances of the CTT cannot be solved within a reasonable time using a MIP solver for the monolithic formulation. Therefore Burke et al. (2010) propose methods to derive lower and upper bounds based on the monolithic formulation. They obtain one lower bound by ignoring the soft constraints **RC** and **RStab**, which gives the possibility to ignore the assigning of rooms in the formulation. They add a constraint to ensure that no more lectures are scheduled in any period than the number of rooms that are available. They note that the constraint is equivalent to aggregating all the rooms into a single room with a capacity equal to the largest room and the number of lectures that can be scheduled in this room in a period is equal to the total number of rooms. They use this observation to generate another lower bound by aggregating the rooms into *multi-rooms*. A multi-room is a set of rooms where the capacity is equal to the capacity of the largest room in the set and the number of lectures that can be scheduled in the multi-room in a period is equal to the number of rooms in the set. In the tests, they show that the latter method provides stronger bounds on average. After they obtain a solution from one of the latter methods a *diving* heuristic is applied by taking the monolithic formulation and fixing the periods based on the solution from the lower bounding method to obtain a full solution. In a more recent paper, Burke et al. (2012) give an exact branch-and-cut algorithm which they also base on the monolithic formulation, but some of the objective costs are left out and instead added as cuts during the solution process. Furthermore, some valid inequalities are presented, which the MIP solver can take advantage of during the search. Their computational results show that the cutting plane approach leads to a better performance.

Lach and Lübbecke (2008) and Lach and Lübbecke (2012) propose a method that divides the CTT into two stages, where they formulate each stage as a MIP model. In the first stage, they schedule the courses into periods, which only requires a binary variable for each course and period. They ensure that all the hard constraints are satisfied in the first stage problem. However only the soft constraints **MWD**, **IL** and **RC** can be taken into account. They include the **RC** constraints in the first stage by adding a variable for each course, period and distinct room capacity. In the second stage, they assign the courses to the rooms that the lectures should take place in, taking the **RStab** constraint into consideration. The solution from the

first stage is used to fix the courses at the determined periods and the selected room capacities in the second stage.

Hao and Benlic (2011) propose a divide-and-conquer approach based on the first stage formulation of Lach and Lübbecke (2012), focusing on finding lower bounds. The method they present divides the MIP model into smaller parts by relaxing or removing some of the constraints such that they can decompose the model into a set of subproblems. They then obtain lower bounds for each subproblem, and the sum of all these lower bounds is then a lower bound of the original problem. The approach provides excellent results.

Cacchiani et al. (2013) presented different formulations containing exponentially many variables. The approach that obtained the best computational results consists of two sets of the main binary decision variables. In each of the sets, a binary variable represents a schedule for an entire week. One of the sets of binary variables takes care of the soft constraints **RC** and **RStab** and the other set considers the soft constraints **MWD** and **IL**. The sum of the lower bounds of each of the set is a lower bound of the original problem. The method shows impressive results and obtains good lower bounds.

In the MIP-based methods found in the literature, it seems that decomposing the problem into smaller parts provides the best results. The downside by many of the methods is that they only provide lower bounds or that they come at the cost of sacrificing the guarantee of optimality, i.e., even if we solve the models to optimality, it does not guarantee that the solution is globally optimal. Based on these observations, the goal of this paper is to exploit the knowledge that decreasing the number of main decision variables provides better performance. Furthermore, we want to maintain the guarantee of optimality, and therefore we look for exact MIP formulations.

We base our work on the work (Bagger et al., 2015) presented at the biennial Multidisciplinary International Scheduling Conference: Theory & Applications (MISTA) 2015 (`http://www.schedulingconference.org/`). There are some slight differences in this article compared to the extended abstract submitted to MISTA. In the abstract, we only discussed one flow formulation, and in this article we present two flow formulations. Furthermore, there were some missing details in the proof that we supplied in the appendix (Bagger et al., 2015, Appendix A Proof of Proposition 2). Therefore the proposition and proof have been changed to overcome this and the resulting mathematical model remains the same. Lastly the results have been updated with our newest test runs and with more data sets.

We organize the paper as follows: In Section 4.2 an intuitive MIP model is presented, based on the main binary variables having three indices. We then show how some of the integer requirements on the variables can be relaxed. In Section 4.3 we present the two formulations based on underlying network flow problems; we base the first on a *Minimum Cost Flow* problem and the second on a *Multi-Commodity Flow* Problem. We present and discuss the computational results in Section 4.4 and lastly some perspectives on the flow formulations are considered in Section 4.5.

Throughout the article, we assume that the reader is familiar with the maximum flow problem, the minimum cut problem, the minimum cost flow problem, and the multi-commodity flow problem as well as the maximum-flow/minimum-cut theorem. We refer to Kleinberg and Tardos (2005) and Ahuja et al. (1993) for details on these problems. Furthermore we define the following for notation purposes:

$$(x)^+ := \max\{x, 0\}$$

We use the notation $\mathbb{Z}^+$ for the set of non-negative integers.

4.2 Three-Index Mixed Integer Programming Formulation

In this section, we present an intuitive MIP formulation of the CTT using three-index binary decision variables. We refer to this formulation throughout the article as the *three-index formulation*.

Let $\mathcal{C}$ be the set of courses, $\mathcal{P}$ be the set of periods and $\mathcal{R}$ be the set of rooms. Furthermore, there are days $\mathcal{D}$, curricula $\mathcal{Q}$, lecturers $\mathcal{L}$, the periods $\mathcal{P}_d \subseteq \mathcal{P}$ that belongs to day $d \in \mathcal{D}$, the courses $\mathcal{C}_q \subseteq \mathcal{C}$ which are part of curriculum $q \in Q$ and the courses $\mathcal{C}_l \subseteq \mathcal{C}$ which are all being taught by lecturer $l \in \mathcal{L}$. For each period $p \in \mathcal{P}$ we define the set Φ_p. The set Φ_p contains the periods that are adjacent to p, i.e. the periods occurring on the same day which is in the period directly before or after the time slot of p.

Let L_c be the number of lectures to be scheduled for course $c \in \mathcal{C}$, C_r be the capacity of room $r \in \mathcal{R}$, S_c be the number of students attending course $c \in \mathcal{C}$ and let $F_{c,p}$ be one if it is allowed to schedule a lecture from course $c \in \mathcal{C}$ in period $p \in \mathcal{P}$ and zero otherwise. Lastly, M_c is the minimum number of days that it is preferred to schedule lectures for course $c \in \mathcal{C}$ in.

Let $x_{c,p,r}$ be a binary variable deciding whether to schedule a lecture from course $c \in \mathcal{C}$ in period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$ or not. $t_{c,d}$ is a binary variable taking value one if course $c \in \mathcal{C}$ has at least one lecture at day $d \in \mathcal{D}$, and zero otherwise. w_c is an integer variable denoting the number of days below the given minimum that course $c \in \mathcal{C}$ has lectures. $z_{c,r}$ is a binary variable taking value one if course $c \in \mathcal{C}$ is scheduled in room $r \in \mathcal{R}$ at least once during the week, and zero otherwise. $s_{q,p}$ is a binary variable taking value one if curriculum $q \in \mathcal{Q}$ has an isolated lecture in period $p \in \mathcal{P}$. Let $W^{\mathbf{RC}}$, $W^{\mathbf{IL}}$, $W^{\mathbf{MWD}}$ and $W^{\mathbf{RStab}}$ be the non-negative weights of the constraints **RC**, **IL**, **MWD** and **RStab** respectively. The three-index formulation is given in the following model:

$$\min\ W^{\mathbf{RC}} \sum_{c\in\mathcal{C},p\in\mathcal{P},r\in\mathcal{R}} (S_c - C_r)^+ \cdot x_{c,p,r} + W^{\mathbf{IL}} \sum_{q\in\mathcal{Q},p\in\mathcal{P}} s_{q,p}$$
$$+ W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c + W^{\mathbf{RStab}} \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}} z_{c,r} - 1 \right) \tag{4.1}$$

$$\text{s.t.} \sum_{p\in\mathcal{P},r\in\mathcal{R}} x_{c,p,r} = L_c \qquad \forall c \in \mathcal{C} \tag{4.2}$$

$$\sum_{r\in\mathcal{R}} x_{c,p,r} \leq F_{c,p} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.3}$$

$$\sum_{c\in\mathcal{C}} x_{c,p,r} \leq 1 \qquad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{4.4}$$

$$\sum_{c\in\mathcal{C}_l,r\in\mathcal{R}} x_{c,p,r} \leq 1 \qquad \forall l \in \mathcal{L}, p \in \mathcal{P} \tag{4.5}$$

$$\sum_{c\in\mathcal{C}_q,r\in\mathcal{R}} x_{c,p,r} \leq 1 \qquad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{4.6}$$

$$\sum_{p\in\mathcal{P}} x_{c,p,r} \leq L_c \cdot z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.7}$$

$$\sum_{p\in\mathcal{P}} x_{c,p,r} \geq z_{c,r} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.8}$$

$$\sum_{r\in\mathcal{R}} z_{c,r} \geq 1 \qquad \forall c \in \mathcal{C} \tag{4.9}$$

$$\sum_{p\in\mathcal{P}_d,r\in R} x_{c,p,r} \geq t_{c,d} \qquad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{4.10}$$

$$\sum_{d\in\mathcal{D}} t_{c,d} + w_c \geq M_c \qquad \forall c \in \mathcal{C} \tag{4.11}$$

$$\sum_{c\in\mathcal{C}_q,r\in\mathcal{R}} \left(x_{c,p,r} - \sum_{p'\in\Phi_p} x_{c,p',r} \right) \leq s_{q,p} \qquad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{4.12}$$

$$x_{c,p,r} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{4.13}$$

$$z_{c,r} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.14}$$

$$t_{c,d} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{4.15}$$

$$w_c \in \mathbb{Z}^+ \qquad \forall c \in \mathcal{C} \tag{4.16}$$

$$s_{q,p} \in \mathbb{B} \qquad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{4.17}$$

The constraints (4.2) ensure that all lectures are scheduled. The courses are only allowed to be scheduled in available periods, which is ensured by the constraints (4.3) that also ensure that all lectures are scheduled in different periods. Constraints (4.4) make sure that no more than one course is scheduled in a room in every period. Constraints (4.5) and (4.6) ensure that courses that share a lecturer or are in the same curriculum are not scheduled in the same period. The constraints (4.7) ensure that lectures are only scheduled in rooms that have been *opened*,

whereby *opened* we mean a room $r \in \mathcal{R}$ where $z_{c,r} = 1$ for course $c \in \mathcal{C}$ Constraints (4.8) and (4.9) ensure that at least one lecture is scheduled in the *open* rooms and that at least one room is put to use by each course. Constraints (4.10) puts an upper bound on the $t_{c,d}$ variable for each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ such that it can take a positive value only if at least one lecture for c is scheduled at day d. Constraints (4.11) calculate the violation of the **MWD** constraint and the constraints (4.12) calculate for which periods the curricula have isolated lectures.

The variables $t_{c,d}$, w_c and $s_{q,p}$ can be relaxed to continuous variables, so we replace the variable domains; (4.15), (4.16) and (4.17) by the following:

$$0 \leq t_{c,d} \leq 1 \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{4.18}$$
$$w_c \geq 0 \quad \forall c \in \mathcal{C} \tag{4.19}$$
$$0 \leq s_{q,p} \leq 1 \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{4.20}$$

Burke et al. (2012) note that it is never possible that the minimum workings days constraint is violated by more than $M_c - 1$ since every course must be scheduled in at least one day. Therefore this can be used as an upper bound for the variable w_c which can help the MIP solver. This bound can be strengthened as all lectures must be scheduled for each course $c \in \mathcal{C}$ and therefore the number of days that the course is scheduled in must be at least $\left\lceil \frac{L_c}{|\mathcal{T}|} \right\rceil$, so we can add the following bounds:

$$0 \leq w_{c,d} \leq M_c - \left\lceil \frac{L_c}{|\mathcal{T}|} \right\rceil \quad \forall c \in \mathcal{C} \tag{4.21}$$

The constraints (4.9) also comes from Burke et al. (2012) where they note that since all lectures must be scheduled then surely every course must occupy at least one room.

If $F_{c,p} = 0$ for some course $c \in \mathcal{C}$ and period $p \in \mathcal{P}$ then we do not add the variables $\{x_{c,p,r}\}_{r \in R}$ to the model. Therefore the constraints (4.3) are redundant since every course is taught by exactly one lecturer and constraints (4.5) ensure that each lecturer has at most one lecture scheduled in any period. Furthermore the constraints (4.5) and (4.6) are replaced by clique inequalities. We do this by creating a graph where each node corresponds to a course. An edge is connecting two courses if they are in the same curriculum or taught by the same lecturer. We then enumerate all the maximal cliques by running the Bron-Kerbosch algorithm (Bron and Kerbosch, 1973). Let Γ be the set of cliques and let C_γ be the set of courses in the clique $\gamma \in \Gamma$. We replace all the constraints (4.5) and (4.6) by adding the following constraints for each clique $\gamma \in \Gamma$ and period $p \in P$:

$$\sum_{c \in C_\gamma, r \in R} x_{c,p,r} \leq 1 \quad \forall \gamma \in \Gamma, p \in P \tag{4.22}$$

4.3 Network Flow Formulations

Our inspiration for the new MIP formulations comes from the two-stage decomposition described by Lach and Lübbecke (2008) and Lach and Lübbecke (2012). In their work, they solve the problem by splitting the MIP model into two distinct models and then solve these

models in sequence. In this paper, we also consider the CTT as two independent models (which are both derived from the three-index model), but instead of solving the two models independently, we re-combine the models into one model using flow formulation techniques. Thereby we obtain new exact formulations with different properties than the original three-index formulation. In Section 4.3.1 we present a formulation for the CTT based on minimum cost flow. In Section 4.3.2 we present a formulation for the CTT based on multi-commodity flow.

4.3.1 Minimum Cost Flow

In this section, we present a formulation based on the minimum cost flow problem. In Section 4.3.1.1 and Section 4.3.1.2 we present two different MIP models for handling different aspects of the problem; the *course-to-period* assignment and the *course-to-room* assignment, respectively. Both of these models are derived from model (4.1) – (4.17). In Section 4.3.1.3 we present how these two models are combined into a single model using minimum flow techniques. The resulting model of this combination has fewer integer variables than the original formulation in model (4.1) – (4.17), and is exact concerning the original formulation of the CTT.

4.3.1.1 Course-to-Period Assignment

In this section we consider the CTT problem from the period aspect only, i.e., we ignore the existence of rooms. The goal of the problem is to assign courses to periods using the same criteria as in the three-index formulation. We give the MIP formulation of the course period assignment subproblem in the following model:

$$\min\ W^{\mathbf{IL}} \sum_{q\in\mathcal{Q},p\in\mathcal{P}} s_{q,p} + W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c \tag{4.23}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}} x_{c,p} = L_c \quad \forall c \in \mathcal{C} \tag{4.24}$$

$$x_{c,p} \leq F_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.25}$$

$$\sum_{c\in\mathcal{C}_\gamma} x_{c,p} \leq 1 \quad \forall \gamma \in \Gamma, p \in \mathcal{P} \tag{4.26}$$

$$\sum_{p\in\mathcal{P}_d} x_{c,p} - t_{c,d} \geq 0 \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{4.27}$$

$$\sum_{d\in\mathcal{D}} t_{c,d} + w_c \geq M_c \quad \forall c \in \mathcal{C} \tag{4.28}$$

$$\sum_{c\in\mathcal{C}_q} \left(x_{c,p} - \sum_{p'\in\Phi_p} x_{c,p'} \right) \leq s_{q,p} \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{4.29}$$

$$x_{c,p} \in \mathbb{B} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.30}$$

$$0 \leq t_{c,d} \leq 1 \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{4.31}$$

$$0 \leq w_c \leq M_c - \left\lceil \frac{L_c}{|\mathcal{T}|} \right\rceil \quad \forall c \in \mathcal{C} \tag{4.32}$$

$$0 \leq s_{q,p} \leq 1 \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{4.33}$$

Note that model (4.23) – (4.33) is equivalent to model (4.1) – (4.17), except that the rooms are ignored. Therefore we do not describe (4.23) – (4.33) in details.

4.3.1.2 Course-to-Room Assignment

In this section we consider the room assignment sub-problem, ignoring the existence of periods. The goal is to assign courses to rooms and the criteria given in the three-index model. We give the formulation for the room assignment in the following:

$$\min \quad W^{\mathbf{RStab}} \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}} z_{c,r} - 1 \right) \tag{4.34}$$

$$\text{s.t.} \quad \sum_{r\in\mathcal{R}} z_{c,r} \geq 1 \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.35}$$

$$z_{c,r} \in \mathbb{B} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.36}$$

Note that in model (4.34) – (4.36) only the room stability objective is considered and not the room capacity. The reason is due to that the binary variable $z_{c,r}$ only identifies whether or not room $r \in \mathcal{R}$ is used by course $c \in C$ and not how many times the course is occupying the room. The room capacity will be taken care of in the later step.

4.3.1.3 Connecting the Models using the Minimum Cost Flow Problem

In this section we show how the minimum cost flow problem connects model (4.23) – (4.33) and model (4.34) – (4.36). Thereby we aim at obtaining a new exact formulation for the CTT with fewer integer variables.

If a solution $\overline{x}$ to model (4.23) – (4.33) and a solution $\overline{z}$ to model (4.34) – (4.36) is given then a new problem emerges; is the combined solution feasible, i.e., is there a feasible mapping from the assigned periods in $\overline{x}$ to the assigned rooms in $\overline{z}$ such that no room is occupied by two courses in the same period, and if so, what is the minimum room capacity penalty to any feasible mapping? A way to check this is to make a bipartite graph and solve a minimum cost maximum matching problem. For every course $c \in \mathcal{C}$ create a node on the left-hand side of the bipartite graph for each period $p \in \mathcal{P}$ that the course has been assigned to, i.e., if $\overline{x}_{c,p} = 1$. For every combination of a room and a period (r, p) create a node on the right-hand side of the graph. For every pair of course period nodes (c, p) and every pair of room-period node (r, p) put an edge between the nodes (c, p) and (r, p) if the course has been assigned to that room, i.e., if $\overline{z}_{c,r} = 1$, and set the weight of the edge to $W^{\mathbf{RC}} (S_c - C_r)^+$. As an example of the matching problem, consider two courses, c_1 and c_2, two periods, p_1 and p_2 and two rooms, r_1 and r_2. Let course c_1 be teaching two lectures assigned in periods p_1 and p_2 and let course c_2 be teaching one lecture assigned in period p_1. Furthermore let c_1 be assigned to rooms r_1 and r_2 and let c_2 be assigned to room r_2. The corresponding bipartite graph is illustrated in Figure 4.1.

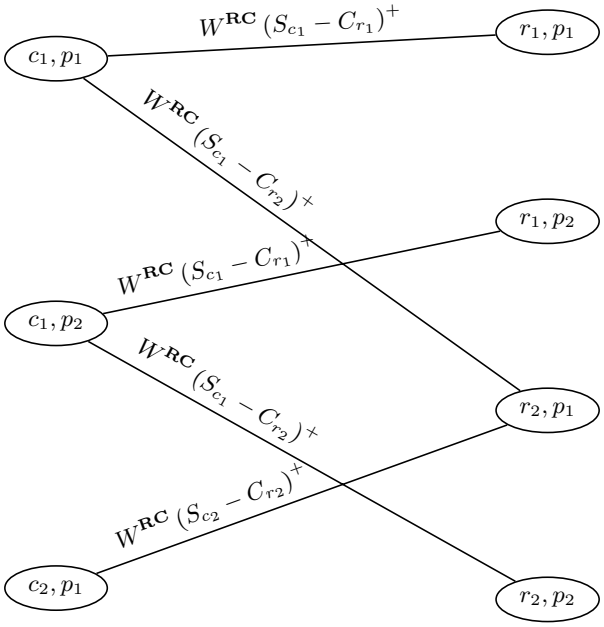


Figure 4.1: Example of the bipartite graph of an instance with courses, c_1 and c_2, periods, p_1 and p_2 and rooms r_1 and r_2. Course c_1 has been assigned to periods p_1 and p_2 and rooms, r_1 and r_2. Course c_2 has been assigned to period p_1 and room r_2. The labels in the nodes indicate the corresponding pair that the node belongs to. The labels above the edges are the corresponding weights.

If the solution $(\overline{x}, \overline{z})$ is feasible, then a maximum matching must match all the left-hand side nodes into a node on the right-hand side, i.e., the value of the maximum matching must

be $\sum_c L_c$ and the cost of the matching corresponds to the total room capacity violation.

A way to solve the minimum cost maximum matching problem is to solve a flow problem on the graph with a source node (u) which is connected to all the left-hand side nodes and a sink node (v) which is connected to all the right-hand side nodes as described by Kleinberg and Tardos (2005) in Chapter 7 and Ahuja et al. (1993) in Section 12.3 and 12.4. The weights on the new edges are all zero and the capacity of all edges are 1. Another way is to create a new graph representation $\mathcal{G}_{\text{mcf}}$ and solve a flow problem on that graph. For every possible course-period combination create a node (c, p) and for every possible room-period combination create a node (r, p). For every course $c \in C$, period $p \in P$ and room $r \in R$ there is an arc from the node (c, p) to the node (r, p). Furthermore there is a source node (u) with an arc to every node (c, p) and a sink node (v) with an arc from every node (r, p). One unit of flow in this graph corresponds to a course assignment, i.e., sending one unit of flow from (u) to (c, p) for some $c \in C$ and $p \in P$ corresponds to assigning course c to period p and sending a unit of flow through the arc $(c, p) \to (r, p)$ for some $c \in C$, $p \in P$ and $r \in R$ corresponds to assigning course c to room r in period p. The capacities are set for a given solution pair of model (4.23) – (4.33) and model (4.34) – (4.36).

For every course $c \in C$ and period $p \in P$ the capacity of the arc $(u) \to (c, p)$ is $\overline{x}_{c,p}$. This is to depict that we can only send flow from the source (u) to a node (c, p) if the course $c \in C$ is assigned to period $p \in P$. For every course $c \in C$, period $p \in P$ and room $r \in R$ the capacity on the arc $(c, p) \to (r, p)$ is $\overline{z}_{c,r}$ to denote that we can only send flow from a course-period pair to a room-period pair if the course is assigned to the corresponding room. The capacity of the arcs $(r, p) \to (v)$ is set to one as a room can at most be occupied by one course in any period. An example of the graph is given in Figure 4.2.

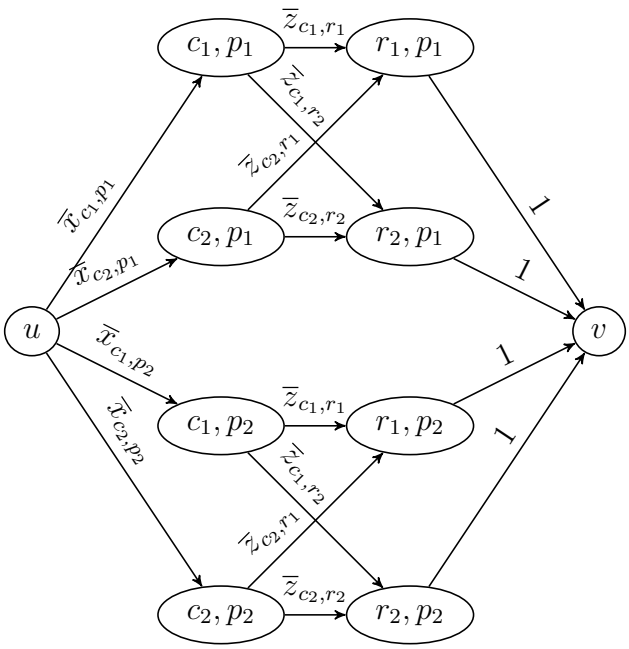


Figure 4.2: Illustration of the minimum cost flow graph of an instance with two courses, two rooms and two period. The labels above the arcs are the corresponding capacities. The weight of the arc from the node (c, p) to the node (r, p) is $W^{\mathbf{RC}}(S_c - C_r)$.

Note that the solutions $\overline{x}$ and $\overline{z}$ are feasible if and only if the solution to the maximum flow in the graph is integral and the total amount (the value) of the flow is $\sum_{c \in C} L_c$. Since we know the amount of flow that any feasible solution must contain we can solve the problem as a minimum cost flow problem where the supply of node (u) and the demand of node (v) is $\sum_{c \in C} L_c$ and all other nodes have a demand and supply of zero. For every course $c \in C$, period $p \in P$ and room $r \in R$ let the cost of the flow on the arc $(c,p) \to (r,p)$ be the cost of violating the **RC** constraint, i.e., $W^{\mathbf{RC}}(S_c - C_r)^+$, and let the cost be zero on all other arcs.

Let the following non-negative variables be defined:

$f^u_{c,p}$: The amount of flow on the arc $(u) \to (c,p)$

$f_{c,p,r}$: The amount of flow on the arc $(c,p) \to (r,p)$

$f^v_{r,p}$: The amount of flow on the arc $(r,p) \to (v)$

The minimum cost flow formulation is given in as follows:

$$\min W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot f_{c,p,r} \tag{4.37}$$

$$\text{s.t.} \quad -\sum_{c \in \mathcal{C}, p \in \mathcal{P}} f^u_{c,p} = -\sum_{c \in \mathcal{C}} L_c \tag{4.38}$$

$$f^u_{c,p} - \sum_{r \in \mathcal{R}} f_{c,p,r} = 0 \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.39}$$

$$\sum_{c \in \mathcal{C}} f_{c,p,r} - f^v_{r,p} = 0 \qquad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{4.40}$$

$$\sum_{r \in \mathcal{R}, p \in \mathcal{P}} f^v_{r,p} = \sum_{c \in \mathcal{C}} L_c \tag{4.41}$$

$$0 \le f^u_{c,p} \le \overline{x}_{c,p} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.42}$$

$$0 \le f_{c,p,r} \le \overline{z}_{c,r} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{4.43}$$

$$0 \le f^v_{r,p} \le 1 \qquad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{4.44}$$

$$f^u_{c,p} \in \mathbb{Z} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.45}$$

$$f_{c,p,r} \in \mathbb{Z} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{4.46}$$

$$f^v_{r,p} \in \mathbb{Z} \qquad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{4.47}$$

The constraints (4.38), (4.39), (4.40) and (4.41) correspond to the balance constraints of the nodes, i.e., ingoing flow minus outgoing flow must equal the demand in the node which is $-\sum_c L_c$ in the source node, $\sum_c L_c$ in the sink node and zero in all the other nodes. Constraints (4.42), (4.43) and (4.44) are the capacity constraints of the arcs.

Model (4.37) – (4.47) contains three-index integer variables just as in model (4.1) – (4.17) which is what we wanted to get rid of to begin with. However, the minimum cost flow integrality theorem states that if all arc capacities, supplies, and demands are integers and there exists a feasible flow, then there exists a minimum cost flow to the LP-relaxation with integer values Ahuja et al. (1993, Theorem 9.10). So we can disregard the integrality requirements on the flow variables since x and z are binary variables.

It should be noted that due to the structure of the graph and the equilibrium constraints (4.39) and (4.40) the flow requirement constraint (4.41) to the sink is redundant and can be removed from the model. Furthermore since $\sum_{p\in P} \overline{x}_{c,p} = L_c$ for every course $c \in C$ in any feasible solution then the flow requirement is fulfilled if and only if equality is met in constraints (4.42). This means that we can remove constraints (4.42) and (4.38) and replace $f^u_{c,p}$ with $\overline{x}_{c,p}$ throughout the model. Lastly since every variable $f^v_{r,p}$ occur exactly once in the constraints (4.40) then we can replace the variable with $\sum_{c\in C} f_{c,p,r}$ in the model. Furthermore we can aggregate all the upper bounds (4.43) into one constraint:

$$\sum_{p\in P} f_{c,p,r} \leq |P| \cdot z_{c,r} \quad \forall c \in C, r \in R \tag{4.48}$$

Since we know that for any course $c \in C$ and room $r \in R$ the sum $\sum_{p\in P} f_{c,p,r}$ can never exceed L_c then we can strengthen this constraint by replacing the coefficient of $z_{c,r}$ with L_c. Then the models (4.23) – (4.33) and (4.34) – (4.36) can be combined into the new formulation given in the following:

$$\begin{aligned}
\min\ & W^{\mathbf{RC}} \sum_{c\in\mathcal{C},p\in\mathcal{P},r\in\mathcal{R}} (S_c - C_r)^+ \cdot f_{c,p,r} \\
& + W^{\mathbf{IL}} \sum_{q\in\mathcal{Q},p\in\mathcal{P}} s_{q,p} + W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c \\
& + W^{\mathbf{RStab}} \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}} z_{c,r} - 1 \right) && (4.49)\\
\text{s.t.}\ & (4.24) \text{ — } (4.33) && (4.50)\\
& (4.35) \text{ — } (4.36) && (4.51)\\
& \sum_{r\in\mathcal{R}} f_{c,p,r} = x_{c,p} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} && (4.52)\\
& \sum_{p\in\mathcal{P}} f_{c,p,r} \leq L_c \cdot z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} && (4.53)\\
& \sum_{c\in\mathcal{C}} f_{c,p,r} \leq 1 \qquad \forall r \in \mathcal{R}, p \in \mathcal{P} && (4.54)\\
& f_{c,p,r} \geq 0 \qquad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} && (4.55)
\end{aligned}$$

Even though the minimum cost flow problem has the integrality property, this does not imply that the $f_{c,p,r}$ variables will take integer values when solving model (4.49) – (4.55). So when the model is solved, we check whether the variables are fractional. If they are, we solve the minimum cost flow problem for the $x_{c,p}$ and $z_{c,r}$ variables by some polynomial algorithm returning an integer solution, e.g. the Cycle-Canceling Algorithm Ahuja et al. (1993, proof of Theorem 9.10, section 9.6).

Throughout the article, we assume that the reader is familiar with the maximum flow problem, the minimum cut problem, the minimum cost flow problem, and the multi-commodity flow problem as well as the maximum-flow/minimum-cut theorem.

4.3.2 Multi-Commodity Flow

In this section, we consider the room assignment part of the problem once again. However, this time we not only formulate a model to decide which rooms the courses should be scheduled in but also how many times during the week the courses should be scheduled in each room. We show how a multi-commodity flow problem can be used to connect this formulation to the time scheduling formulation in model (4.23) – (4.33).

We introduce the integer variable $y_{c,r}$ to identify the number of times that course $c \in C$ is assigned to room $r \in R$. The formulation is given as follows:

$$\min\ W^{\mathbf{RC}} \cdot \sum_{c \in \mathcal{C}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot y_{c,r} + W^{\mathbf{RStab}} \cdot \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) \tag{4.56}$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R}} y_{c,r} = L_c \qquad \forall c \in \mathcal{C} \tag{4.57}$$

$$y_{c,r} \leq L_c \cdot z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.58}$$

$$y_{c,r} \geq z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.59}$$

$$\sum_{r \in \mathcal{R}} z_{c,r} \geq 1 \qquad \forall c \in \mathcal{C} \tag{4.60}$$

$$y_{c,r} \in \mathbb{Z}^+ \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.61}$$

$$z_{c,r} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.62}$$

Constraints (4.57) ensure that the total number of times that a course $c \in C$ is occupying some rooms is equal to the number of lectures to be taught. Constraints (4.58) ensure that for some course $c \in C$ and some room $r \in R$, $z_{c,r}$ is set to one if $y_{c,r} > 0$. Constraints (4.59) and (4.60) ensure that at least one room is selected for each course and that at least one lecture is put into each selected room.

Given a solution $\overline{x}$ to model (4.23) – (4.33) and a solution $(\overline{y}, \overline{z})$ to model (4.56) – (4.62) then a new problem emerges; is the combined solution $(\overline{x}, \overline{y}, \overline{z})$ feasible, i.e., is there a feasible mapping from the assigned rooms in $\overline{y}$ to the assigned periods in $\overline{x}$ such that no room is occupied by two courses in the same period and no course is giving two lectures in the same period? To check this, we formulate the problem as a *multi-commodity flow problem*. For each course $c \in \mathcal{C}$ we have a commodity where we need to send flow from a source node (c^-) to a sink node (c^+). The demand for the commodity is L_c, i.e., the amount of flow that needs to be sent from the source to the sink is the number of lectures for the course. For each period $p \in \mathcal{P}$ we have a node and for each room $r \in \mathcal{R}$ we also have a node. For each course $c \in \mathcal{C}$ we have an outgoing arc from node (c^-) to each node (p) with a capacity of $\overline{x}_{c,p}$ for the corresponding period $p \in \mathcal{P}$. The capacity ensures that the amount of flow send out of the node (c^-) to the node (p) does not exceed $\overline{x}_{c,p}$, i.e., a course can only send flow to (be scheduled in) the periods that it has been assigned to. Since each commodity has their distinct source node, this ensures that the amount of the commodity corresponding to course c is only send to periods where the course is assigned. Furthermore we have an ingoing arc from each node (r) to node (c^+) with a

capacity of $\overline{y}_{c,r}$ for the corresponding room $r \in \mathcal{R}$. The capacity ensures that the total amount of flow through rooms does not exceed the number of times the course has been scheduled in the rooms. Lastly for each period and room pair $(p, r) \in \mathcal{P} \times \mathcal{R}$ there is an arc from node (p) to node (r) with a capacity of one to ensure that the total amount of flow across all commodities does not exceed one. This capacity corresponds to the constraint that any room cannot be occupied by more than one lecture in any period. An example of the graph for a test instance containing two courses, two rooms and two periods is given in Figure 4.3.

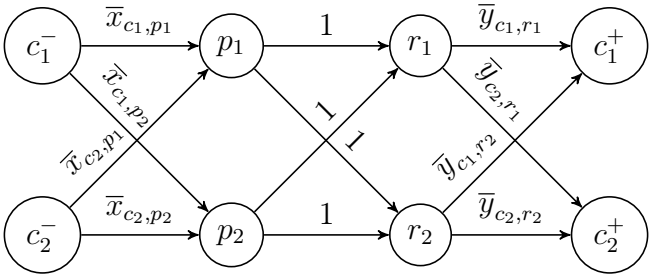


Figure 4.3: Illustration of the multi-commodity flow graph of an instance with two courses, two rooms and two periods. The labels above the arcs are the corresponding capacities.

We can either state the multi-commodity flow problem as an arc formulation or a path formulation. We use the path formulation. In the graph it can be seen that there is a path $(c_1^-) \to (p) \to (r) \to (c_2^+)$ for every course $c_1 \in \mathcal{C}$, every period $p \in \mathcal{P}$, every room $r \in \mathcal{R}$ and every course $c_2 \in \mathcal{C}$. These are all the paths in the graph and each commodity corresponding to a course $c \in \mathcal{C}$ must select L_c paths to send flow. The path $(c_1^-) \to (p) \to (r) \to (c_2^+)$, for some course $c_1 \in \mathcal{C}$, some period $p \in \mathcal{P}$, some room $r \in \mathcal{R}$ and some course $c_2 \in \mathcal{C}$, can only be selected by a commodity corresponding to course $c \in \mathcal{C}$ if $c = c_1 = c_2$. Let the integer variable $f_{c,p,r}$ correspond to the amount of flow of the commodity corresponding to course $c \in \mathcal{C}$ that is send on the path $(c^-) \to (p) \to (r) \to (c^+)$ for period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$. Then the mathematical formulation can be described as follows:

$$\sum_{r \in \mathcal{R}} f_{c,p,r} \leq \overline{x}_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.63}$$

$$\sum_{c \in \mathcal{C}} f_{c,p,r} \leq 1 \quad \forall p \in \mathcal{P}, r \in \mathcal{R} \tag{4.64}$$

$$\sum_{p \in \mathcal{P}} f_{c,p,r} \leq \overline{y}_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.65}$$

$$\sum_{p \in \mathcal{P}, r \in \mathcal{P}} f_{c,p,r} = L_c \quad \forall c \in \mathcal{C} \tag{4.66}$$

$$f_{c,p,r} \in \mathbb{Z}^+ \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{4.67}$$

Constraints (4.63), (4.64) and (4.65) ensure that the capacities of the arcs are not violated. Constraint (4.66) ensures that the demand is fulfilled for each commodity and (4.67) ensures that the flow is integral.

The multi-commodity flow problem is known to be NP-hard, however, in the next Proposition 4.1 we show that it suffices to solve the LP-relaxation of the path formulation in model (4.63) – (4.67).

Proposition 4.1. *Let A be the set of feasible period-room solutions and let $o(\overline{x}, \overline{y}, \overline{z})$ denote the objective value of the (possibly infeasible) period-room solution $(\overline{x}, \overline{y}, \overline{z})$. Consider the solution space of the LP-relaxation $\mathcal{F}_{LP}$ of model (4.63) – (4.67) for $(\overline{x}, \overline{y}, \overline{z})$. We then have the following:*

$$\begin{aligned}
&(a) \quad \mathcal{F}_{LP} = \emptyset \implies (\overline{x}, \overline{y}, \overline{z}) \notin A \\
&(b) \quad \mathcal{F}_{LP} \neq \emptyset \implies \exists y' : o(\overline{x}, y', \overline{z}) \in A \wedge \\
&\qquad\qquad\qquad\qquad\qquad\qquad o(\overline{x}, y', \overline{z}) \leq o(\overline{x}, \overline{y}, \overline{z})
\end{aligned}$$

Proposition 4.1a. We make the proof by showing that the contrapositive statement holds:

$$(\overline{x}, \overline{y}, \overline{z}) \in A \implies \mathcal{F}_{LP} \neq \emptyset$$

Assume that $(\overline{x}, \overline{y}, \overline{z}) \in A$ and consider some feasible assignment for this solution. Let the variable $\overline{f}_{c,p,r}$ take value one if course $c \in \mathcal{C}$ is assigned to period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$ in the considered assignment. Since we are considering a feasible assignment and it is based on the solution $(\overline{x}, \overline{y}, \overline{z})$ then the following conditions must be met:

$$\sum_{r \in \mathcal{R}} \overline{f}_{c,p,r} = \overline{x}_{c,p} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.68}$$

$$\sum_{c \in \mathcal{C}} \overline{f}_{c,p,r} \leq 1 \qquad \forall p \in \mathcal{P}, r \in \mathcal{R} \tag{4.69}$$

$$\sum_{p \in \mathcal{P}} \overline{f}_{c,p,r} = \overline{y}_{c,r} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.70}$$

$$\overline{f}_{c,p,r} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{4.71}$$

So the variables $\overline{f}$ must fullfil the constraints (4.63)–(4.65) and (4.67). Furthermore since $\sum_{p \in \mathcal{P}} \overline{x}_{c,p} = \sum_{r \in \mathcal{R}} \overline{y}_{c,r} = L_c$ for each course $c \in \mathcal{C}$ then $\sum_{p \in \mathcal{P}, r \in \mathcal{R}} \overline{f}_{c,p,r} = L_c$ must hold and thus constraint (4.66) is also fulfilled. This means that $\overline{f}$ must be a feasible solution to model (4.63) – (4.67) and so $\mathcal{F}_{LP} \neq \emptyset$. □

Proposition 4.1b. Consider a solution $(\overline{x}, \overline{y}, \overline{z})$ with objective value $o(\overline{x}, \overline{y}, \overline{z})$. Assume that $\mathcal{F}_{LP} \neq \emptyset$. Then there must exist some (possibly fractional) solution $\overline{f}$ that fulfills the constraints in model (4.63) – (4.67).
Since $\sum_{p \in \mathcal{P}} \overline{x}_{c,p} = \sum_{r \in \mathcal{R}} \overline{y}_{c,r} = L_c$ for every course $c \in \mathcal{C}$ then the only way to fulfill constraints (4.66) is to fulfill the constraints (4.63) and (4.65) by equality, i.e., the solution $\overline{f}$ must fulfill the following conditions:

$$\sum_{r\in\mathcal{R}} \overline{f}_{c,p,r} = \overline{x}_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{4.72}$$

$$\sum_{c\in\mathcal{C}} \overline{f}_{c,p,r} \leq 1 \quad \forall p \in \mathcal{P}, r \in \mathcal{R} \tag{4.73}$$

$$\sum_{p\in\mathcal{P}} \overline{f}_{c,p,r} = \overline{y}_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{4.74}$$

$$\overline{f}_{c,p,r} \geq 0 \quad \forall c \in \mathcal{C}, p \in \mathcal{C}, r \in \mathcal{R} \tag{4.75}$$

Now we create a new solution $(\overline{x}, y', \overline{z})$ where the values $\overline{x}$ and $\overline{z}$ are fixed and only the values of the y-variables might be changed while ensuring that the objective value does not increase,
i.e., $o(\overline{x}, y', \overline{z}) \leq o(\overline{x}, \overline{y}, \overline{z})$. Note that since $\overline{x}$ and $\overline{z}$ are kept fixed the difference in the objective value of the two solutions $(\overline{x}, \overline{y}, \overline{z})$ and $(\overline{x}, y', \overline{z})$ must be on the **RC** constraint.

Two cases can occur; either $\overline{f}$ is integral, or some of the values are fractional. If $\overline{f}$ is integral then every value must be either zero or one due to constraint (4.64) and therefore $(\overline{x}, \overline{y}, \overline{z}) \in A$ and we simply set $y' = \overline{y}$.

Consider now the case where $\overline{f}$ contains fractional values. Since $\overline{y}_{c,r} \leq L_c \cdot \overline{z}_{c,r}$ for each course $c \in \mathcal{C}$ and each room $r \in \mathcal{R}$ then we must have that $\overline{f}_{c,p,r} \leq \overline{z}_{c,r}$ for every period $p \in \mathcal{P}$. Consider now the minimum cost flow problem as described in Section 4.3.1. A flow $\overline{f}^{\text{MCF}}$ in the minimum cost flow graph is created in the following way; for each course $c \in \mathcal{C}$, period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$ send flow on the path $(u) \to (c,p) \to (r,p) \to (v)$ equal to $\overline{f}_{c,p,r}$. All the node balancing constraints must be satisfied as we are considering paths from the source to the sink. Furthermore the amount of flow must be $\sum_{c\in\mathcal{C}} L_c$ since $\sum_{p\in\mathcal{P}} \overline{x}_{c,p} = L_c$ for each course $c \in \mathcal{C}$ and the condition (4.72) holds. Due to the condition (4.72) the capacity of the arc $(u) \to (c,p)$ in the minimum cost flow graph illustrated in Figure 4.2 cannot be violated as the total amount of flow on the arc is equal to $\sum_{r\in\mathcal{R}} \overline{f}_{c,p,r}$ and the capacity of the arc is $\overline{x}_{c,p}$. For each course $c \in \mathcal{C}$, period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$ the capacity of the arc $(c,p) \to (r,p)$, which is $\overline{z}_{c,r}$, cannot be violated since the flow send through that arc is equal to $\overline{f}_{c,p,r}$ and we just argued that $\overline{f}_{c,p,r} \leq \overline{z}_{c,r}$. Lastly since $\sum_{c\in\mathcal{C}} \overline{f}_{c,p,r} \leq 1$ then the capacity of the arc $(r,p) \to (v)$ in the graph from Figure 4.2 for room $r \in \mathcal{R}$ and period $p \in \mathcal{P}$ cannot be violated. So the flow $\overline{f}^{\text{MCF}}$ is feasible for the minimum cost flow graph. Recall that the costs on the arcs in the minimum cost flow graph is zero on the arc $(u) \to (c,p)$ for each course $c \in \mathcal{C}$ and period $p \in \mathcal{P}$, zero on the arc $(r,p) \to (v)$ for each room $r \in \mathcal{R}$ and period $p \in \mathcal{P}$ and $W^{\mathbf{RC}} \cdot (S_c - C_r)^+$ on the arc $(c,p) \to (r,p)$ for each course $c \in \mathcal{C}$, period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$. This means that the total cost of the flow $\overline{f}^{\text{MCF}}$ is:

$$W^{\mathbf{RC}} \sum_{c\in\mathcal{C}, p\in\mathcal{P}, r\in\mathcal{R}} (S_c - C_r)^+ \cdot \overline{f}_{c,p,r}$$

Due to the integrality property of the minimum cost flow there must exists a flow f' where the total amount of flow is the same as for the flow $\overline{f}^{\text{MCF}}$, the flow on all the arcs are integers and the cost of the flow f' must be less than or equal to the cost of the flow $\overline{f}^{\text{MCF}}$:

$$W^{\mathbf{RC}} \sum_{c\in\mathcal{C}, p\in\mathcal{P}, r\in\mathcal{R}} (S_c - C_r)^+ \cdot f'_{c,p,r} \leq W^{\mathbf{RC}} \sum_{c\in\mathcal{C}, p\in\mathcal{P}, r\in\mathcal{R}} (S_c - C_r)^+ \cdot \overline{f}_{c,p,r} \tag{4.76}$$

Create the new solution $(\overline{x}, y', \overline{z})$ by setting $y' = \sum_{p \in \mathcal{P}} f'_{c,p,r}$. Now the difference in the objective value between the solution $(\overline{x}, y', \overline{z})$ and the solution $(\overline{x}, \overline{y}, \overline{z})$ can be calculated:

$$\begin{aligned} o(\overline{x}, y', \overline{z}) - o(\overline{x}, \overline{y}, \overline{z}) =& W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot y'_{c,r} \\ & - W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot \overline{y}_{c,r} \\ =& W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot f'_{c,p,r} \\ & - W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot \overline{y}_{c,r} \end{aligned}$$

Using (4.76) we can put an upper bound on the difference in the objective values:

$$o(\overline{x}, y', \overline{z}) - o(\overline{x}, \overline{y}, \overline{z}) \leq W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot \left(\sum_{p \in \mathcal{P}} \overline{f}_{c,p,r} - \overline{y}_{c,r} \right)$$

Due to condition (4.74) it must therefore hold that $o(\overline{x}, y', \overline{z}) \leq o(\overline{x}, \overline{y}, \overline{z})$. □

Proposition 4.1 shows that we only need the linear relaxation of the multi-commodity flow problem and thus do not need to require integrality. Any solution to model (4.63) – (4.67) can only fulfill constraint (4.66) if equality is met in constraints (4.65). Then model (4.23) – (4.33) and model (4.56) – (4.62) can be combined into the following model:

$$\begin{aligned} \min\ & W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R}} (S_c - C_r)^+ \cdot f_{c,p,r} \\ & + W^{\mathbf{IL}} \sum_{q \in \mathcal{Q}, p \in \mathcal{P}} s_{q,p} + W^{\mathbf{MWD}} \sum_{c \in \mathcal{C}} w_c \\ & + W^{\mathbf{RStab}} \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) && (4.77) \\ \text{s.t.}\ & (4.24) \text{ — } (4.33) && (4.78) \\ & (4.57) \text{ — } (4.62) && (4.79) \\ & \sum_{p \in \mathcal{P}} f_{c,p,r} = y_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} && (4.80) \\ & \sum_{r \in \mathcal{R}} f_{c,p,r} \leq x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} && (4.81) \\ & \sum_{c \in \mathcal{C}} f_{c,p,r} \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} && (4.82) \\ & f_{c,p,r} \geq 0 \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} && (4.83) \end{aligned}$$

It is not guaranteed that the $f_{c,p,r}$ variables are integers in the solution obtained from model (4.77) – (4.83). If the solution returned by the model contains fractional values for the $f_{c,p,r}$

variables we solve a minimum cost flow problem in the same way as in Section 4.3.1 using the values of the variables $x_{c,p}$ and $z_{c,p}$. The minimum cost flow gives us a feasible integer solution with an objective value which is less than or equal to the objective value of the solution returned by model (4.77) – (4.83) as we showed in the proof of Proposition 4.1.

4.4 Computational Results

In this section, we perform computational experiments to evaluate the performance of the two new formulations for the CTT. We test the models on three data sets; TEST, COMP and DDS. The set TEST contains four data instances and were proposed by Di Gaspero and Schaerf (2003) (test1 – test4). The set COMP contains 21 data instances from the ITC2007 competition track 3 described in Di Gaspero et al. (2007) (comp01 – comp21) mainly taken from the University of Udine. The set DDS contains seven data instances mainly taken from other Italian universities (DDS1 – DDS7). All the data sets can be retrieved from `http://tabu.diegm.uniud.it/ctt/index.php`. A benchmarking tool was provided as part of the ITC2007 competition, which calculates the amount of time that the algorithms were allowed to run in the competition on the machine the tool is executed on. This amount of time is usually referred to as one CPU unit. The tool can be obtained from `http://www.cs.qub.ac.uk/itc2007`. We ran the tests in Windows 10 on a 3.40GHz Intel® Core™ i5-3570K CPU with 8GB memory. Running the benchmarking tool returned 208 seconds as one CPU unit. The MIP solver used is Gurobi 6.0.2 provided by Gurobi Optimization Inc. (2015). We have run all tests with the default parameters except that we have set the presolver to the most aggressive level (Presolve=2), in the hope that the presolver can decrease the size of the problem, and we set the number of threads to one (Threads=1).

As mentioned in sections 4.3.1.3 and 4.3.2, it may be needed to run some minimum cost flow algorithm on the solutions returned by model (4.49) – (4.55) or model (4.77) – (4.83). However, in all our tests the final solutions did not contain any fractional variables, so the minimum cost flow algorithm was never put to use. We also mentioned in Section 4.2 that an algorithm for enumerating all clique inequalities was run. This algorithm takes less than a second even for the largest data instances we have tested, so we have neglected these enumerations from the time limits when solving the models.

4.4.1 Lower Bounds Results

The bounds obtained by the flow formulations are compared with the four approaches proposed by Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011) and Cacchiani et al. (2013). In Table 4.1 – 4.3 we report the lower bounds obtained on the first 14 COMP data sets for the latter mentioned four approaches and the flow-based formulations when running the approaches for one CPU unit, ten CPU units and forty CPU units. In each table, we report the number of times that the approaches obtain a bound which is at least as good as the bound obtained by the other approaches and also the number of times that the approaches obtain a bound which is better than all the other approaches. For each data instance, we rank the approach according to the bound obtained. The approach that obtains the highest bound on a data instance gets rank one; the second highest gets ranked two, and so on. If multiple approaches are tied then each approach is assigned the average of the ranks, e.g., if

three approaches are tied for rank two, three and four then they are each assigned the rank three as this is the average of the three values. We report the average of the ranks over all the instances for each approach.

Table 4.1: Comparison of the lower bounds obtained for the different approaches when given one CPU time unit; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013), MIN (the minimum cost flow based formulation) and MULT (the multi-commodity flow based formulation). The numbers reported in bold font are the values where the specific models obtained a value which is at least as good as the other formulations. The numbers underlined are the values where the specific models obtained a value which is better than for the other formulations. The two second last lines (Best) denotes the number of times that a specific algorithm obtained a lower bound which was at least as good as the other algorithms (in bold font) and strictly better than the other algorithms (underlines in bold font). The last line (Rank) reports the average rank obtained by the specific algorithm.

Instance	BMPR10	LL12	HB11	CCRT13	MIN	MULT
comp01	0	4	4	**5**	**5**	**5**
comp02	0	0	**10**	0	0	6
comp03	25	0	26	24	**27**	26
comp04	**35**	22	**35**	**35**	24	24
comp05	119	92	19	6	**132**	121
comp06	**13**	7	12	0	12	12
comp07	**6**	0	5	0	0	0
comp08	**37**	30	**37**	**37**	26	27
comp09	68	37	39	**92**	49	46
comp10	3	2	**4**	0	**4**	**4**
comp11	**0**	**0**	**0**	**0**	**0**	**0**
comp12	**101**	29	43	0	85	85
comp13	52	33	46	**57**	39	38
comp14	**41**	40	**41**	32	40	**41**
Best	**2**	**1**	**6**	**6**	**5**	**4**
	3	**0**	**1**	**2**	**2**	**0**
Rank	**2.71**	4.89	2.96	3.96	3.29	3.18

In Table 4.1 –4.3 we see that the proposed formulations can compete with most of the approaches, except for the proposed method by Cacchiani et al. (2013) which seems to perform better on most instances. The performance of Cacchiani et al. (2013) is especially evident in Table 4.4 where we compare all the three data sets between the flow formulations and the approach proposed by Cacchiani et al. (2013) where the time limit is forty CPU units. Here we see that Cacchiani et al. (2013) obtains a better bound more often than the flow formulations. However the flow formulations appear to generate a better bound on two of the instances at ten and forty CPU units; comp05 and comp12, where the bound generated for comp12 is an improvement of the best-known bound.

Table 4.2: Comparison of the lower bounds obtained for the different approaches when given ten CPU time units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013), MIN (the minimum cost flow based formulation) and MULT (the multi-commodity flow based formulation). The interpretation of the numbers in bold font and the underlined numbers follows that of Table 4.1.

Instance	BMPR10	LL12	HB11	CCRT13	MIN	MULT
comp01	4	4	4	**5**	**5**	**5**
comp02	0	8	12	**16**	8	8
comp03	33	0	34	**52**	37	35
comp04	**35**	28	**35**	**35**	**35**	**35**
comp05	111	25	69	6	**173**	172
comp06	**15**	10	12	11	13	13
comp07	**6**	2	**6**	**6**	0	**6**
comp08	**37**	34	**37**	**37**	**37**	**37**
comp09	65	41	67	**92**	71	71
comp10	**4**	**4**	**4**	2	**4**	**4**
comp11	**0**	**0**	**0**	**0**	**0**	**0**
comp12	95	32	78	0	**129**	116
comp13	52	39	53	**57**	54	54
comp14	42	41	43	**48**	43	42
Best	**6**	**2**	**5**	**10**	**7**	**6**
	1	**0**	**0**	**5**	**2**	**0**
Rank	3.75	5.18	3.46	3.00	**2.75**	2.86

4.4.2 Comparing Upper-Bound Formulations

Since Lach and Lübbecke (2012) and Burke et al. (2010) obtain both lower and upper bounds we also compare these with the bounds obtained by the flow formulations. The results for forty CPU units are given in Table 4.5. Here we see that the flow formulations obtain better lower bounds in most cases. As for the upper bounds, Lach and Lübbecke (2012) and the minimum cost flow formulation appear to perform similarly. The minimum cost flow formulation obtains upper bounds that are at least as good as the other formulations in most cases, but Lach and Lübbecke (2012) obtain upper bounds that are better than the other formulations in more cases. However, the minimum cost flow formulation has the best average performance, both regarding upper and lower bounds, as we see it has a lower average rank than any of the other formulations.

Table 4.3: Comparison of the lower bounds obtained for the different approaches when given forty CPU time units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013), MIN (the minimum cost flow based formulation) and MULT (the multi-commodity flow based formulation). The interpretation of the numbers in bold font and the underlined numbers follows that of Table 4.1.

Instance	BMPR10	LL12	HB11	CCRT13	MIN	MULT
comp01	**5**	4	4	**5**	**5**	**5**
comp02	1	11	12	**16**	8	8
comp03	33	25	36	**52**	38	37
comp04	**35**	28	**35**	**35**	**35**	**35**
comp05	114	108	80	166	**186**	181
comp06	**16**	10	**16**	11	**16**	**16**
comp07	**6**	**6**	**6**	**6**	0	**6**
comp08	**37**	**37**	**37**	**37**	**37**	**37**
comp09	66	46	67	**92**	74	73
comp10	**4**	**4**	**4**	2	**4**	**4**
comp11	**0**	**0**	**0**	**0**	**0**	**0**
comp12	95	53	84	100	**142**	140
comp13	54	41	55	57	56	**59**
comp14	42	46	43	**48**	44	43
Best	**7**	**4**	**6**	**9**	**8**	**8**
	0	**0**	**0**	**4**	**2**	**1**
Rank	4.00	4.61	3.82	**2.75**	2.89	2.93

Table 4.4: Comparison of the lower bounds obtained for the different model formulations when given forty CPU time units; CCRT13 (Cacchiani et al., 2013), MIN (the minimum cost flow based formulation) and MULT (the multi-commodity flow based formulation). The interpretation of the numbers in bold font and the underlined numbers follows that of Table 4.1.

Instance	CCRT13	MIN	MULT
comp01	**5**	**5**	**5**
comp02	**16**	8	8
comp03	**52**	38	37
comp04	**35**	**35**	**35**
comp05	166	**186**	181
comp06	11	**16**	**16**
comp07	**6**	0	**6**
comp08	**37**	**37**	**37**
comp09	**92**	74	73
comp10	2	**4**	**4**
comp11	**0**	**0**	**0**
comp12	100	**142**	140
comp13	57	56	**59**
comp14	**48**	44	43
comp15	**52**	38	37
comp16	**13**	**13**	11
comp17	**48**	43	44
comp18	**52**	36	30
comp19	48	**56**	55
comp20	**4**	0	0
comp21	**68**	56	57
DDS1	40	**46**	44
DDS2	**0**	**0**	**0**
DDS3	**0**	**0**	**0**
DDS4	**17**	15	15
DDS5	**0**	**0**	**0**
DDS6	**0**	**0**	**0**
DDS7	**0**	**0**	**0**
test1	**224**	**224**	**224**
test2	**16**	**16**	**16**
test3	**59**	**59**	**59**
test4	**46**	44	43
Best	**25**	**19**	**16**
	11	**4**	**1**
Rank	**1.81**	2.00	2.19

Table 4.5: Comparison of the lower and upper bound bounds obtained for the different model formulations when given forty CPU time units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), MIN (the minimum cost flow based formulation) and MULT (the multi-commodity flow based formulation). The interpretation of the numbers in bold font and the underlined numbers follows that of Table 4.1.

	BMPR10		LL12		MIN		MULT	
Instance	LB	UB	LB	UB	LB	UB	LB	UB
comp01	**5**	9	4	12	**5**	**5**	**5**	**5**
comp02	1	63	**11**	46	8	**45**	8	59
comp03	33	123	25	**66**	**38**	123	37	99
comp04	**35**	36	28	38	**35**	**35**	**35**	**35**
comp05	114	629	108	368	**186**	**355**	181	377
comp06	**16**	**46**	10	51	**16**	92	**16**	92
comp07	**6**	45	**6**	**25**	0	179	**6**	-
comp08	**37**	41	**37**	44	**37**	**37**	**37**	41
comp09	66	105	46	**99**	**74**	105	73	103
comp10	**4**	23	**4**	**16**	**4**	18	**4**	68
comp11	**0**	12	**0**	7	**0**	**0**	**0**	**0**
comp12	95	785	53	548	**142**	**423**	140	500
comp13	54	67	41	66	56	66	**59**	**59**
comp14	42	55	**46**	**53**	44	55	43	74
Best	**7**	**1**	**6**	**5**	**10**	**7**	**8**	**4**
	0	**1**	**2**	**5**	**4**	**4**	**1**	**1**
Rank	2.75	3.14	3.11	2.25	**2.00**	**2.07**	2.14	2.53

4.4.3 Comparing with the Three-Index Formulation

In this section, we compare the flow formulations with the three-index formulation.

In Table 4.6 we provide the results of both the three-index formulation (model (4.1) – (4.17)) and the flow based formulations. Here we see that the flow formulations clearly outperform the three-index formulation and we obtain a new lower bound in one of the instances compared to the best-known bound. This improvement makes the models riveting as some of the other approaches based on the three-index formulation from the literature might also benefit from this reformulation.

On a final note, these computational results show that the the minimum cost flow based formulation in general has better performance than the multi-commodity flow based formulation as it obtains a much lower average rank.

Table 4.6: Comparison of the lower and upper bound bounds obtained for the different model formulations when given forty CPU time units; 3IDX (the three-index formulation), MIN (the minimum cost flow based formulation) and MULT (the multi-commodity flow based formulation). The interpretation of the numbers in bold font and the underlined numbers follows that of Table 4.1.

	3IDX		MIN		MULT	
Instance	LB	UB	LB	UB	LB	UB
comp01	**5**	**5**	**5**	**5**	**5**	**5**
comp02	0	109	**8**	**45**	**8**	59
comp03	33	136	**38**	123	37	**99**
comp04	**35**	41	**35**	**35**	**35**	**35**
comp05	161	427	**186**	**355**	181	377
comp06	12	98	**16**	**92**	**16**	**92**
comp07	3	**118**	0	179	**6**	-
comp08	**37**	45	**37**	**37**	**37**	41
comp09	66	157	**74**	105	73	**103**
comp10	**4**	39	**4**	**18**	**4**	68
comp11	**0**	**0**	**0**	**0**	**0**	**0**
comp12	108	629	**142**	**423**	140	500
comp13	51	126	56	66	**59**	**59**
comp14	41	144	**44**	**55**	43	74
comp15	33	136	**38**	123	37	**99**
comp16	8	102	**13**	61	11	**42**
comp17	41	175	43	123	**44**	**109**
comp18	24	133	**36**	**78**	30	108
comp19	53	114	**56**	**57**	55	**57**
comp20	**0**	146	**0**	**50**	**0**	96
comp21	49	235	56	156	**57**	**133**
DDS1	45	92	**46**	**70**	44	76
DDS2	**0**	**0**	**0**	**0**	**0**	**0**
DDS3	**0**	**0**	**0**	**0**	**0**	**0**
DDS4	**15**	67	**15**	**17**	**15**	12079
DDS5	**0**	**0**	**0**	**0**	**0**	**0**
DDS6	**0**	51	**0**	**2**	**0**	27
DDS7	**0**	**0**	**0**	**0**	**0**	**0**
test1	**224**	233	**224**	**224**	**224**	**224**
test2	**16**	**19**	**16**	**19**	**16**	**19**
test3	**59**	83	**59**	**75**	**59**	**75**
test4	43	109	**44**	**91**	43	107
Best	**15**	**8**	**28**	**24**	**21**	**19**
	0	**1**	**11**	**12**	**4**	**7**
Rank	2.45	2.66	**1.66**	**1.55**	1.89	1.80

4.5 Perspectives

We proposed two mixed integer programming models for the curriculum based course timetabling problem (CTT). We based both our models on network flow problems; the minimum cost flow problem and the multi-commodity flow problem. These models are exact, meaning that they will obtain optimal solutions given enough computational resources. We showed that these models have far fewer integer variables than the standard three-index formulation for the CTT. By experimental results, we also showed that the models outperform the three-index formulation regarding finding feasible solutions when solved by a MIP solver.

Regarding lower bounds, the formulations are competitive with most of the mixed integer programming based approaches from the literature and improve one of the currently best known lower bounds on the benchmarking instances from the second international timetabling competition.

Regarding upper bounds, the minimum cost flow based formulation performs better than other state-of-the-art MIP based approaches.

Furthermore, some of the approaches from the literature are based on models similar to the three-index formulation. Since we showed that our formulations outperform the three-index formulation, we believe that these approaches can also benefit from adopting the proposed network flow formulations.

Acknowledgements

The authors would like to thank Professor Stephan Røpke, Department of Management Engineering, Technical University of Denmark, and Professor Carsten Thomassen, Department of Applied Mathematics and Computer Science, Technical University of Denmark for fruitful discussions on the graphs and proofs. Niels-Christian Fink Bagger's industrial PhD project is funded by Innovation Fund Denmark (IFD). IFD has supported the work solely financially and has not participated in any research related activities

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0-13-617549-X.

Bagger, N., Kristiansen, S., Sørensen, M., and Stidsen, T. (2015). "Flow Formulation-based Model for the Curriculum-based Course Timetabling Problem". In: *MISTA 2015 Proceedings*. Ed. by Z. Hanzálek, G. Kendall, B. McCollum, and P. Šøucha. Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2015), pp. 825–848.

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). "An overview of curriculum-based course timetabling". In: *TOP*, pp. 1–37.

Bonutti, A., De Cesco, F., Di Gaspero, L., and Schaerf, A. (2012). "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results". In: *Annals of Operations Research* 194.1, pp. 59–70.

Bron, C. and Kerbosch, J. (1973). "Algorithm 457: Finding All Cliques of an Undirected Graph". In: *Commun. ACM* 16.9, pp. 575–577. ISSN: 0001-0782. DOI: 10.1145/362342.362367. URL: http://doi.acm.org/10.1145/362342.362367.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2008). "Penalising Patterns in Timetables: Novel Integer Programming Formulations". In: *Operations Research Proceedings 2007*. Ed. by J. Kalcsics and S. Nickel. Vol. 2007. Operations Research Proceedings. 10.1007/978-3-540-77903-2_63. Springer Berlin Heidelberg, pp. 409–414. ISBN: 978-3-540-77903-2.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010). "A supernodal formulation of vertex colouring with applications in course timetabling". In: *Annals of Operations Research* 179.1, pp. 105–130.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). "A branch-and-cut procedure for the Udine Course Timetabling problem". In: *Annals of Operations Research* 194.1, pp. 71–87.

Cacchiani, V., Caprara, A., Roberti, R., and Toth, P. (2013). "A new lower bound for curriculum-based course timetabling". In: *Computers and Operation Research* 40.10, pp. 2466–2477. DOI: 10.1016/j.cor.2013.02.010.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). *The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)*. Tech. rep. School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom.

Di Gaspero, L. and Schaerf, A. (2003). "Multi-neighbourhood Local Search with Application to Course Timetabling". English. In: *Practice and Theory of Automated Timetabling IV*. Ed. by E. Burke and P. De Causmaecker. Vol. 2740. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 262–275. ISBN: 978-3-540-40699-0. DOI: 10.1007/978-3-540-45157-0_17. URL: http://dx.doi.org/10.1007/978-3-540-45157-0_17.

Gurobi Optimization Inc. (2015). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Hao, J. K. and Benlic, U. (2011). "Lower bounds for the ITC-2007 curriculum-based course timetabling problem". In: *European Journal of Operational Research* 212.3, pp. 464–472.

Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321295358.

Lach, G. and Lübbecke, M. (2012). "Curriculum based course timetabling: new solutions to Udine benchmark instances". In: *Annals of Operations Research* 194, pp. 255–272. ISSN: 0254-5330.

Lach, G. and Lübbecke, M. E. (2008). "Optimal University Course Timetables and the Partial Transversal Polytope". In: *International Workshop on Experimental and Efficient Algorithms*. Springer, pp. 235–248.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., and Burke, E. K. (2010). "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition". In: *INFORMS Journal on Computing* 22.1, pp. 120–130.

5 Benders' Decomposition for Curriculum-based Course Timetabling

Niels-Christian F. Bagger[a,b] · Matias Sørensen[a,b] · Thomas R. Stidsen[a]

[a]mORetime research group, Management Science, Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 426B, DK-2800 Kgs. Lyngby, Denmark, http://www.moretime.man.dtu.dk/

[b]MaCom A/S, Vesterbrogade 48, 1., DK-1620 København V, Denmark

Status: Submitted to Computers & Operations Research

Abstract: In this paper we apply Benders' decomposition to the Curriculum-based Course Timetabling (CTT) problem. The objective of the CTT problem is to assign a set of lectures to time slots and rooms. Our approach is based on splitting the problem into a time scheduling problem and a room allocation problem. The Benders' algorithm is then used to generate cuts that connect the time schedule and room allocation. We only generate feasibility cuts, meaning that most of the solutions we obtain from a mixed integer programming solver are infeasible, and therefore we also provide a heuristic to regain feasibility.

We compare our algorithm with other approaches from the literature on a total of 32 data instances. We get a lower bound on 23 of the instances which are at least as good as the lower bounds obtained by the state-of-the-art, and on eight of these our lower bounds are higher. On two of the instances, our lower bound is an improvement of the currently best-known. Lastly, we compare our decomposition with the model without the decomposition on additionally six instances which are much larger than the other 32. To our knowledge, this is the first time that lower bounds are calculated for these six instances.

Keywords: University Course Timetabling · Integer Programming · Benders' Decomposition · Maximum Flow · Minimum Cut

5.1 Curriculum-based Course Timetabling

In this work we consider the Curriculum based Course Timetabling Problem (CTT) introduced in track 3 of the second international timetabling competition (ITC2007) as described by Di

Gaspero et al. (2007), McCollum et al. (2010) and Bonutti et al. (2012). Most of the work on CTT focuses on finding high-quality solutions using heuristics. The drawback of these heuristics is that they do not provide any proof of quality, e.g., how far from optimality the solutions are. We need bounding and exact methods to be able to validate the quality of the heuristics and not much work has been put into developing these methods. In this article we apply Benders' decomposition to a Mixed Integer Programming (MIP) model that we presented in Bagger et al. (2016).

In the CTT problem we must schedule weekly lectures for multiple courses into time periods and assign the lectures to rooms. We are given a set of days, each divided into a set of time slots. We refer to a day and time slot combination as a period. The basic entities of the problem are the *courses* to schedule, and the *periods* and *rooms* that are available. The problem originates from a real world application and has received a lot of attention since the competition. Each course has a number of lectures which must all be scheduled in a period and assigned a room. Furthermore all the lectures must be scheduled in distinct periods. This requirement is referred to as the Lectures (**L**) constraint. The courses have specified some of the periods to be *unavailable* periods, i.e., periods where it is not allowed to schedule the course. This requirement is referred to as the Availability (**A**) constraint. There are no constraints on assigning the courses to rooms, i.e., any course can be assigned to any room. Besides the courses, periods and rooms the problem also contains *lecturers* and *curricula*, hence the name *Curriculum-based Course Timetabling*. Each course is taught by a lecturer, and a curriculum is a set of courses which may be followed by the same students. If two courses are taught by the same lecturer or belong to the same curriculum they cannot have lectures scheduled in the same periods. This requirement is referred to as the Conflicts (**C**) constraint. For each room, at most one lecture can be assigned in any period, which is referred to as the Room Occupancy (**RO**) constraint. The objective of the CTT is to find a timetable which fulfils all the latter mentioned requirements, **L**, **A**, **C** and **RO**, while minimizing a weighted sum of the violation of four soft constraints; Room Capacity (**RC**), Room Stability (**RStab**), Minimum Working Days (**MWD**) and Isolated Lectures (**IL**). When a course is assigned to a room where the number of seats is smaller than the number of students that are attending the course then the constraint **RC** is violated by one for each student above the capacity of the room. It is desired to assign lectures from the same course to as few distinct rooms as possible. A course is violating constraint **RStab** by the total number of distinct rooms that it is assigned to minus one. The constraint **MWD** is the desire to spread lectures across a given number of days. We say that a day is a working day for a course if at least one lecture from the course is scheduled in a time slot on that day. For each course a number of minimum working days is provided and if the number of working days is below this number then the violation of the constraint is the difference. The last constraint **IL** is associated with the curricula. For each curriculum it is desired to have as few *isolated* lectures as possible. Each *isolated* lecture counts as one violation. A curriculum has an *isolated* lecture in a period if any of the courses belonging to the curriculum has a lecture scheduled in the period and none of the courses have lectures scheduled in the *adjacent* periods. We say that two periods are adjacent if they belong to the same day and are in consecutive time slots.

In the following section 5.1.1 we provide an overview of other approaches applied to CTT. In section 5.2 we first provide a brief introduction to Benders' decomposition and then we describe how we apply it to CTT. In section 5.3 we describe a heuristic to repair partially infeasible

solutions, whereby partially infeasible we mean solutions where the time schedule is feasible, but not the room assignment. In section 5.4 we describe the computational results. Lastly, in section 5.5 we state our conclusions on this work.

5.1.1 Related Research

As we consider a MIP model for the problem, we mainly focus on other MIP based approaches in the literature. For a thorough overview of the problem and different approaches for CTT we refer to Bettinelli et al. (2015).

Burke et al. (2008) and Burke et al. (2010) introduces a compact MIP formulation that is exact, in the sense that the optimal solution can be found by a generic MIP solver given enough computational resources. However, many instances of the CTT cannot be solved for this formulation within a reasonable time using a MIP solver and so Burke et al. (2010) propose methods to derive lower and upper bounds. They obtain lower bounds by aggregating the rooms into *multi-rooms*. For each *multi-room* the number of lectures that can be scheduled in it in any period is equal to the umber of rooms that where aggregated. This problem provides a lower bound for CTT. To obtain an upper bound they fix the periods (or parts of them) according to the solution from the lower bounding mechanism and then assign rooms to the lectures. Burke et al. (2012) give an exact branch-and-cut algorithm which they also base on the compact formulation, but some of the objective costs are left out and instead added as cuts during the solution process. This can be seen as a Benders' decomposition, but instead of generating the cuts dynamically they are generated a priori and then added when needed.

Lach and Lübbecke (2008) and Lach and Lübbecke (2012) proposed a method that divides the CTT into two stages. The started by grouping the rooms together such that if two rooms have the same capacity then they are in the same group. Then in the first stage, they scheduled the courses into periods and assigned them to these capacities. This method is a Benders' Decomposition (Lübbecke, 2015). In the second stage, they assign the courses to the rooms, where the solution from the first stage is used to fix the courses at the determined periods and the selected room capacities.

Hao and Benlic (2011) divides the MIP model that Lach and Lübbecke (2012) use in the first-stage into smaller parts by relaxing or removing some of the constraints. The relaxations makes it possible to decompose the model into a set of subproblems where they calculate a lower bound for each subproblem. The sum of all these lower bounds is then a lower bound for CTT. Cacchiani et al. (2013) present multiple extended MIP formulation, i.e., models with an exponential amount of variables. The approach that provides the best results divides the problem into two parts; one focusing on the time scheduling related soft constraints and the other focusing on the room related soft constraints. They calculate a lower bound for each part and the sum of these lower bounds is then a lower bound for CTT.

In Bagger et al. (2016) two MIP models were presented that were inspired by Lach and Lübbecke (2008) and Lach and Lübbecke (2012) and Burke et al. (2008) and Burke et al. (2010). The division of the problem described Lach and Lübbecke (2012) by was applied (excluding the notion of distinct capacities) and it was shown that the two stages could be connected using two underlying flow network formulations. The first formulation was based on a minimum cost flow network and performed best of the two. The second formulation was based on a multi-commodity flow problem which is the formulation where we apply Benders' decomposition

(Benders, 1962) in this article. The reason for using the last formulation, although it did not perform as well as the first, is that the underlying network is a feasibility problem. So we do not need to generate any optimality cuts, but only feasibility cuts.

5.2 Benders' Decomposition

In this section we give an introduction to Benders' decomposition followed by our application of the technique. Our introduction is a crude overview and we refer to (Benders, 1962) and Martin (1999, chapter 10) for a detailed description. We describe the method based on a model containing two types of variables, x and y. The x variables are non-negative continuous variables, and we do not have any assumptions on the y variables, i.e., $x \geq 0$ and $y \in Y$ where Y can be any domain, e.g., the set of integers. Consider the MIP model (5.1).

$$\begin{aligned} \min \quad & c^\top x + f(y) \\ \text{s.t.} \quad & Ax + B(y) \geq b \\ & y \in Y \\ & x \geq 0 \end{aligned} \tag{5.1}$$

In model (5.1) $c \in \mathbb{R}^n$ is the cost vector of the x variables, $A \in \mathbb{R}^{n \times m}$ is the constraint matrix of the x variables and $b \in \mathbb{R}^m$ is the right-hand-side vector of the constraints. $f : Y \to \mathbb{R}$ is some function to evaluate the cost of the y variables and B is a vector function that evaluates the contribution of the y variables for the constraints. If we fix the y variables to some value in the domain Y then what remains is a linear programme (LP). This assumption can be extended as described by Geoffrion (1972), but we stick to the (LP) case in this context. Model (5.1) can be rewritten to model (5.2).

$$\begin{aligned} \min \quad & f(y) + z \\ \text{s.t.} \quad & z \geq \min_{x \geq 0} \left\{ c^\top x \mid Ax \geq b - B(y) \right\} \\ & y \in Y \\ & z \in \mathbb{R} \end{aligned} \tag{5.2}$$

In model (5.2) there is an inner optimization problem in the constraints. If the y variables are fixed, then this is an LP and we can change it into its dual LP as in model (5.3).

$$\begin{aligned} \min \quad & f(y) + z \\ \text{s.t.} \quad & z \geq \max_{\pi \geq 0} \left\{ (b - B(y))\, \pi \mid A^\top \pi \leq c \right\} \\ & y \in Y \\ & z \in \mathbb{R} \end{aligned} \tag{5.3}$$

One interesting aspect of the inner optimization problem in model (5.3) is that the corresponding polytope is independent of the values of the y variables. So if the polytope $\left\{ A^T \pi \leq c \right\}$ is non-empty then we can reformulate the problem using the extreme points Π^p and extreme rays Π^r as in model (5.4).

$$
\begin{aligned}
\min \quad & f(y) + z \\
\text{s.t.} \quad & z \geq (b - B(y))\,\overline{\pi}^p \quad \forall \overline{\pi}^p \in \Pi^p \\
& 0 \geq (b - B(y))\,\overline{\pi}^r \quad \forall \overline{\pi}^r \in \Pi^r \\
& y \in Y \\
& z \in \mathbb{R}
\end{aligned} \tag{5.4}
$$

Model (5.4) is referred to as Benders' master problem. For a given solution $\overline{y}$ model (5.5) is referred to as Benders' subproblem.

$$
\begin{aligned}
\max \quad & (b - B(\overline{y}))\,\pi \\
\text{s.t.} \quad & A^\top \pi \leq c \\
& \pi \geq 0
\end{aligned} \tag{5.5}
$$

As the number of extreme points and rays can be exponentially large, a way to solve the model is to relax the master problem by removing some (or all) of the constraints originating from the extreme points and rays and then iteratively add them as needed. This is done by finding a solution $\overline{y}$ for the master problem (5.4) and inserting the solution into the subproblem (5.5). The subproblem is then solved to obtain an extreme point $\overline{\pi}^p$ or ray $\overline{\pi}^r$ and the corresponding cut is added to the master problem if it is violated. This is done iteratively as illustrated in Figure 5.1.

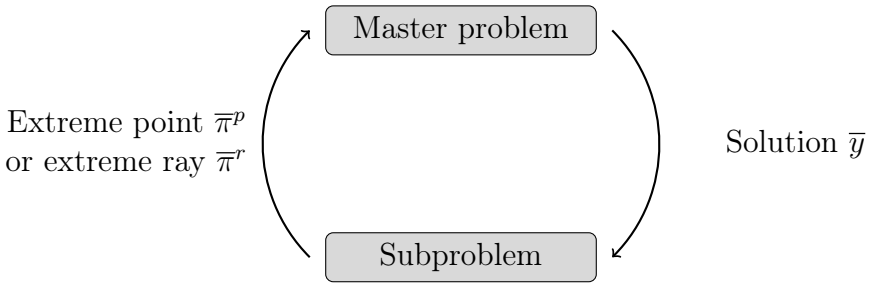


Figure 5.1: The iterative loop of Benders' algorithm.

A lower bound on the (relaxed) master problem is a lower bound on the original model, and if the subproblem returns an extreme point, then this provides an upper bound. The iterative process is run until some stopping criterion is met, e.g., a time limit is reached, or the lower and upper bounds are sufficiently close.

In the following section 5.2.1 we give the model we use for the decomposition and Benders' master problem. In section 5.2.2 we state the subproblems we use to generate Benders' cuts. For all the models we are given the set of courses $\mathcal{C}$, the set of periods $\mathcal{P}$ and the set of rooms $\mathcal{R}$. We are also provided with the set of days $\mathcal{D}$ and the set of periods $\mathcal{P}_d$ belonging to day $d \in \mathcal{D}$. For each period $p \in \mathcal{P}$ the set θ_p is the set of periods that are adjacent to p, i.e., the periods that belong to the same day and are in consecutive time slots. Furthermore, we have the set of lecturers $\mathcal{L}$ and curricula $\mathcal{Q}$. For each curriculum $q \in \mathcal{Q}$ we have the set of courses $\mathcal{C}_q$ belonging to the curriculum. We also define the set Γ which is the set of course-cliques. A course-clique $\gamma \in \Gamma$ is a set of courses $\mathcal{C}_\gamma$ such that for each period $p \in \mathcal{P}$ at most one lecture from $\mathcal{C}_\gamma$ can be scheduled in that period. We find the set of course-cliques in the same way as Bagger

et al. (2016). For each course $c \in \mathcal{C}$ we have the number of lectures that needs to be scheduled $L_c \in \mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of non-negative integers, and the minimum number of requested working days $M_c \in \mathbb{Z}^+$. We also have the number of students attending the course $S_c \in \mathbb{Z}^+$ and for each period $p \in \mathcal{P}$ the parameter $F_{c,p} \in \mathbb{B}$ specifying whether or not it is feasible to schedule c in p. For each room, we have the parameter $C_r \in \mathbb{Z}^+$ which is the capacity of the room. We have $W^{\mathbf{MWD}} \in \mathbb{R}^+$, $W^{\mathbf{IL}} \in \mathbb{R}^+$, $W^{\mathbf{RC}} \in \mathbb{R}^+$ and $W^{\mathbf{RStab}} \in \mathbb{R}^+$ which are the non-negative weights of the soft constraints, **MWD**, **IL**, **RC** and **RStab** respectively. Lastly, for any value $x \in \mathbb{R}$ we use the notation $(x)^+$ to denote that we are taking the maximum of x and zero.

5.2.1 Master Problem

The model we use for the decomposition is the multi-commodity flow formulation that we presented in Bagger et al. (2016). We start by describing this model, and then we describe how we reformulate it using Benders' decomposition.

For each course $c \in \mathcal{C}$ and period $p \in \mathcal{P}$ there is a binary variable $x_{c,p}$ which is set to one if c is scheduled in p and zero otherwise. To calculate the violation of the requested minimum working days we need to calculate which days the courses are scheduled to. We define a binary variable $t_{c,d}$ for each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ which is set to one if c is scheduled in at least one of the periods $\mathcal{P}_d$ and zero otherwise. Let w_c for each course $c \in \mathcal{C}$ be an integer variable counting the number of days, below the minimum requested, that c has lectures scheduled. Lastly for each curriculum $q \in \mathcal{Q}$ and period $p \in \mathcal{P}$ there is a binary variable $s_{q,p}$ that is set to one if q has an isolated lecture in p and zero otherwise. We can then formulate the time scheduling part of the problem as follows:

$$\sum_{p \in \mathcal{P}} x_{c,p} = L_c \quad \forall c \in \mathcal{C} \tag{5.6}$$

$$x_{c,p} \leq F_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{5.7}$$

$$\sum_{c \in \mathcal{C}_\gamma} x_{c,p} \leq 1 \quad \forall \gamma \in \Gamma, p \in \mathcal{P} \tag{5.8}$$

$$\sum_{p \in \mathcal{P}_d} x_{c,p} \geq t_{c,d} \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{5.9}$$

$$\sum_{d \in \mathcal{D}} t_{c,d} + w_c \geq M_c \quad \forall c \in \mathcal{C} \tag{5.10}$$

$$\sum_{c \in \mathcal{C}_q} \left(x_{c,p} - \sum_{p' \in \theta_p} x_{c,p'} \right) \leq s_{q,p} \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{5.11}$$

$$x_{c,p} \in \mathbb{B} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{5.12}$$

$$t_{c,d} \in \mathbb{B} \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{5.13}$$

$$w_c \in \mathbb{Z}^+ \quad \forall c \in \mathcal{C} \tag{5.14}$$

$$s_{q,p} \in \mathbb{B} \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{5.15}$$

Constraints (5.6) and (5.7) ensure that all lectures are scheduled, that they are scheduled in different periods and that they are only scheduled in the periods where the courses are available.

Constraints (5.8) ensure that at most one lecture from each course-clique is scheduled in every period. The violation of the **MWD** constraint is calculated in (5.9) and (5.10). Lastly, the constraints (5.11) calculate in which periods the curricula have isolated lectures. The domains of the variables are given in (5.12) – (5.15).

For the room assignment part of the problem we define an integer variable $y_{c,r}$ for each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ which counts the number of times that c has been assigned to r. $z_{c,r}$ is a binary variable that is set to one if course $c \in \mathcal{C}$ is assigned to room $r \in \mathcal{R}$ at least once and zero otherwise. The room assignment part can then be formulated as follows:

$$\sum_{r \in \mathcal{R}} y_{c,r} = L_c \qquad \forall c \in \mathcal{C} \tag{5.16}$$

$$y_{c,r} \leq L_c z_{c,r} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.17}$$

$$y_{c,r} \geq z_{c,r} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.18}$$

$$\sum_{r \in \mathcal{R}} z_{c,r} \geq 1 \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.19}$$

$$y_{c,r} \in \mathbb{Z}^+ \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.20}$$

$$z_{c,r} \in \mathbb{B} \qquad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.21}$$

Constraints (5.16) ensure that all lectures is assigned a room. For every course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ the variable $y_{c,r}$ must be set to zero if $z_{c,r}$ is zero and it must be set to a positive value if $z_{c,r}$ is set to one. This is ensured by the constraints (5.17) and (5.18). Since every lecture must be scheduled, then every course must be assigned to at least one room which is reflected by the constraints (5.19). The domains of the variables are given in (5.20) – (5.21).

Finally, from the two parts; (5.6) – (5.15) and (5.16) – (5.21), we can formulate the objective function which is to be minimized:

$$\begin{aligned} o(x,y,z) := & W^{\mathbf{MWD}} \sum_{c \in \mathcal{C}} w_c \\ & + W^{\mathbf{RC}} \sum_{c \in \mathcal{C}, r \in \mathcal{R}} (S_c - C_r)^+ y_{c,r} \\ & + W^{\mathbf{RStab}} \sum_{c \in \mathcal{C}} \left(\sum_{r \in \mathcal{R}} z_{c,r} - 1 \right) \\ & + W^{\mathbf{IL}} \sum_{q \in \mathcal{Q}, p \in \mathcal{P}} s_{q,p} \end{aligned} \tag{5.22}$$

Until now, we have given the time scheduling and room assignment part of the problem as separate parts. Given a binary variable $f_{c,p,r}$ for each course $c \in \mathcal{C}$, period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$, which is set to one if c is scheduled in room r at period p, the two parts can be connected as follows:

$$\sum_{p \in \mathcal{P}} f_{c,p,r} = y_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.23}$$

$$\sum_{r \in \mathcal{R}} f_{c,p,r} \leq x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{5.24}$$

$$\sum_{c \in \mathcal{C}} f_{c,p,r} \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{5.25}$$

$$f_{c,p,r} \in \mathbb{B} \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{5.26}$$

We show in Bagger et al. (2016) that the integrality requirements (5.26) can be relaxed to non-negativity constraints such that the f variables are continuous. This means that we can project them out and replace (5.23) – (5.26) by the following Benders' cuts:

$$\sum_{c \in \mathcal{C}, p \in \mathcal{P}} \alpha^k_{c,p} x_{c,p} + \sum_{c \in \mathcal{C}, r \in \mathcal{R}} \beta^k_{c,r} y_{c,r} \leq \eta^k \quad \forall k \in \mathcal{K} \tag{5.27}$$

The set $\mathcal{K}$ is the set of Benders' cuts. For each cut $k \in \mathcal{K}$ the coefficient of the variable $x_{c,p}$ is $\alpha^k_{c,p}$ for each course $c \in \mathcal{C}$ and $p \in \mathcal{P}$, and for each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ the coefficient of the variable $y_{c,r}$ is $\beta^k_{c,r}$. The parameter η^k is a constant for each $k \in \mathcal{K}$. Since $\mathcal{K}$ can be exponentially large we add the cuts dynamically when needed. We add some of the cuts statically. For instance, we cannot schedule more courses to a single period than the number of rooms available:

$$\sum_{c \in \mathcal{C}} x_{c,p} \leq |\mathcal{R}| \quad \forall p \in \mathcal{P} \tag{5.28}$$

Similarly the total number of times that some room is assigned to by the courses cannot exceed the total number of periods:

$$\sum_{c \in \mathcal{C}} y_{c,r} \leq |\mathcal{P}| \quad \forall r \in \mathcal{R} \tag{5.29}$$

The cuts (5.28) and (5.29) are added to the master problem before we start the solution process, and we generate the remaining cuts when an integer solution violates them.

5.2.2 Subproblems

In this section, we describe the subproblems. Our first subproblem is based on the problem connecting the x and y variables (5.23) – (5.26). These cuts are necessary and sufficient to ensure that the solutions obtained in our decomposed model are equivalent to the solutions in our non-decomposed model. We base the second subproblem on the indirect connection between the x and z variables. The applicability of the second subproblem is to generate cuts on the z variables as they are connected with the y in the constraints (5.17) and (5.18) so the cuts will also have an indirect affect on the y variables.

5.2.2.1 Dual LP

The first subproblem we use, directly follows from the LP relaxation of (5.23) – (5.26), as it was mentioned before that the integrality constraints can be neglected. Given a solution $(\overline{x}, \overline{y}, \overline{z})$ we fix the x and y variables and if (5.23) – (5.26) is infeasible then we need to identify a cut. It can be verified that due to constraints (5.6) and (5.16) if we remove constraints (5.25) then the problem is feasible. So to find out if (5.23) – (5.26) is infeasible for the given solution, we subtract a non-negative variable u from the left-hand-side of each of the constraints (5.25). We then search for the solution that minimizes u:

$$\min \quad u \tag{5.30}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}} f_{c,p,r} = y_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.31}$$

$$\sum_{r \in \mathcal{R}} f_{c,p,r} = x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{5.32}$$

$$\sum_{c \in \mathcal{C}} f_{c,p,r} - u \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{5.33}$$

$$f_{c,p,r} \geq 0 \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{5.34}$$

$$u \geq 0 \tag{5.35}$$

Note that we have changed the inequality (5.24) into the equality (5.32). This is because any solution $\left(\overline{x}, \overline{y}, \overline{z}, \overline{f}\right)$ that is feasible must meet equality in the constraints (5.24), since $L_c = \sum_{r \in \mathcal{R}} \overline{y}_{c,r} = \sum_{p \in \mathcal{P}, r \in \mathcal{R}} \overline{f}_{c,p,r} \leq \sum_{p \in \mathcal{P}} \overline{x}_{c,p} = L_c$ must hold for each $c \in \mathcal{C}$. The problem (5.23) – (5.26) is feasible for the given solution $(\overline{x}, \overline{y}, \overline{z})$ if and only if there exists a solution for (5.30) – (5.35) where the variables x and y are fixed at the values $\overline{x}$ and $\overline{y}$ respectively and where the value of u is zero. We can, therefore, use this problem to generate feasibility cuts.

Given a solution $(\overline{x}, \overline{y}, \overline{z})$, we fix the variables x and y in (5.30) – (5.35) to the values $\overline{x}$ and $\overline{y}$, i.e., setting the bounds $\overline{x} \leq x \leq \overline{x}$ and $\overline{y} \leq y \leq \overline{y}$, and solve the problem to optimality. Let $\overline{r}^x_{c,p}$ and $\overline{r}^y_{c,r}$ be the reduced costs of the variables x and y respectively and let $\overline{u}$ be the value of the variable u. According to Fischetti (2015) we can derive the Benders' cut as follows:

$$\sum_{c \in \mathcal{C}, p \in \mathcal{P}} \overline{r}^x_{c,p} \left(x_{c,p} - \overline{x}_{c,p}\right) + \sum_{c \in \mathcal{C}, r \in \mathcal{R}} \overline{r}^y_{c,r} \left(y_{c,r} - \overline{y}_{c,r}\right) + \overline{u} \leq 0 \tag{5.36}$$

We add the cut (5.36) whenever it is violated by an integer solution $(\overline{x}, \overline{y}, \overline{z})$.

5.2.2.2 Maximum Flow / Minimum Cut

In the previous section we described how to obtain the cuts from (5.23) – (5.26). Those cuts only connect the x and y variables together. In this section, we look at the link between the x and z variables. Though they are not directly connected in any constraints, they have an indirect relation since the y and z variables are contained in some of the same constraints. Consider the model (5.23) – (5.26). It can be shown that the constraints (5.24) must be met by equality in any feasible solution. When changing the inequality in (5.24) into an equality,

then the equality in (5.23) can be changed to a less-than-or-equal inequality. Combining this with constraint (5.17) we have that:

$$\sum_{p \in \mathcal{P}} f_{c,p,r} \leq L_c z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{5.37}$$

The consequence of (5.37) is that for some course $c \in \mathcal{C}$ and some room $r \in \mathcal{R}$, if $z_{c,r}$ is zero then $f_{c,p,r}$ must be zero for each period $p \in \mathcal{P}$. This leads to the following model:

$$f_{c,p,r} \leq z_{c,r} \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{5.38}$$

$$\sum_{r \in \mathcal{R}} f_{c,p,r} = x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{5.39}$$

$$\sum_{c \in \mathcal{C}} f_{c,p,r} \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{5.40}$$

$$f_{c,p,r} \geq 0 \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{5.41}$$

Note that model (5.38) – (5.41) is decomposable by the periods, i.e., to find out if the model is feasible we can consider one period at a time. We can formulate this problem as a maximum flow problem. Given the solution $(\overline{x}, \overline{y}, \overline{z})$ construct a graph $\overline{\mathcal{G}}_p = \left(\overline{\mathcal{V}}_p, \overline{\mathcal{A}}_p\right)$ for each period $p \in \mathcal{P}$. The graph contains a source node (u) and a sink node (v). Furthermore there is a node (c) for each course $c \in \mathcal{C}$ and a node (r) for each room $r \in \mathcal{R}$. For each course $c \in \mathcal{C}$ there is an arc $(u, c) \in \overline{\mathcal{A}}_p$ where the capacity is set to $\overline{x}_{c,p}$. For each room $r \in \mathcal{R}$ there is an arc $(r, v) \in \overline{\mathcal{A}}_p$ with a capacity of one. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ there is an arc $(c, r) \in \overline{\mathcal{A}}_p$ where the capacity is set to $\overline{z}_{c,r}$. An example of the graph, for an instance with two courses and two rooms for some period $p \in \mathcal{P}$, is illustrated in Figure 5.2.

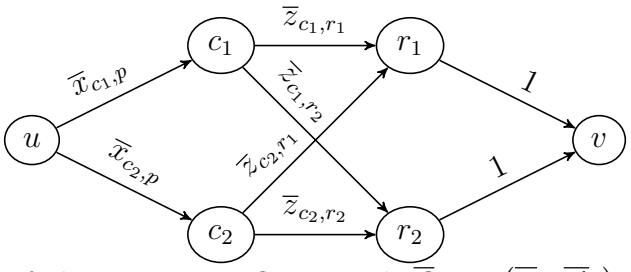


Figure 5.2: Illustration of the maximum flow graph $\overline{\mathcal{G}}_p = \left(\overline{\mathcal{V}}_p, \overline{\mathcal{A}}_p\right)$ for period $p \in \mathcal{P}$ of an instance with two courses and two rooms to validate the $(\overline{x}, \overline{z})$ pair. The labels above the arcs are the corresponding capacities.

As we state in the next proposition 5.1, we can use this graph to check whether or not (5.38) – (5.41) is feasible for a given solution.

Proposition 5.1. *The model (5.38) – (5.41) is feasible for a given solution $(\overline{x}, \overline{y}, \overline{z})$ if and only if for each period $p \in \mathcal{P}$ there exists some flow in graph $\overline{\mathcal{G}}_p$ where the value is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$*

Proof of proposition 5.1. Consider the solution $(\overline{x}, \overline{y}, \overline{z})$ and the graph $\overline{\mathcal{G}}_p$ for some period $p \in \mathcal{P}$. We first prove that if the model is feasible for the period, then there exists some flow in the graph where the value of the flow is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$. We then prove that if there exists some flow in the graph where the value of the flow is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ then there is a feasible solution for the model for the period.

Consider some solution $\overline{f}$ to the model (5.38) – (5.41) for $(\overline{x}, \overline{y}, \overline{z})$. Consider the graph $\overline{\mathcal{G}}_p$ for some period $p \in \mathcal{P}$. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ the amount of flow we send on the path $(u) \to (c) \to (r) \to (v)$ is equal to $\overline{f}_{c,p,r}$. To validate that this is a flow for the graph we need to check if the node balancing constraints and the capacity constraints are met. Since we are only sending flow on paths from the source to the sink, then the node balancing constraints must be met. The flow leaving node (c) is equal to $\sum_{r \in \mathcal{R}} \overline{f}_{c,p,r}$ and due to the node balancing constraints and the constraints (5.39) then the flow on the arc (u, c) must be equal $\overline{x}_{c,p}$ which is the capacity of that arc. The flow on the arc (c, r) is $\overline{f}_{c,p,r}$ and due to constraint (5.38) this is less than or equal to $\overline{z}_{c,r}$ which is the capacity on the arc. The total amount of flow entering node (r) is equal to $\sum_{c \in \mathcal{C}} \overline{f}_{c,p,r}$ and due to the node balancing constraints then the flow on the arc (r, v) must be equal to this value. Because of the constraint (5.40) the flow on the arc (r, v) must, therefore, be less than or equal to one which is the capacity of the arc. The value of the flow must be equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ because of the constraints (5.39). So if (5.38) – (5.41) is feasible for the given solution then there is a flow for the graph where the value of the flow is $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$.

Now consider some flow for the graph $\overline{\mathcal{G}}_p$ for period $p \in \mathcal{P}$ where the value of the flow is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ let $\overline{f}_{c,p,r}$ denote the amount of flow on the arc (c, r). $\overline{f}$ is a feasible solution for the f variables if the constraints (5.38) – (5.40) are not violated. Since the capacity on the arc (c, r) for course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ is $\overline{z}_{c,r}$ then the constraints (5.38) cannot be violated. Since the sum of the capacities leaving the source node is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ then this means that the flow on arc (u, c) for each course $c \in \mathcal{C}$ must be equal to $\overline{x}_{c,p}$ and due to the node balancing constraints then all the flow leaving (c) must also equal this value, i.e., $\sum_{r \in \mathcal{R}} \overline{f}_{c,p,r}$ must be equal to $\overline{x}_{c,p}$, meaning that the constraints (5.39) are fulfilled. The total amount of flow entering node (r) must be equal to $\sum_{c \in \mathcal{C}} \overline{f}_{c,p,r}$ and due to the node balancing constraints this value cannot exceed one, which means that the constraints (5.40) are not violated. So if there exists some flow for the graph where the value of the flow is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ then the model is feasible.

Since we considered an arbitrary period, then this concludes our proof of proposition 5.1. □

For the source node (u) and sink node (v) a $u - v$ cut is a cut that separates the graph into two parts; the u side containing the source node, and the v side containing the sink node. For the maximum flow problem, the value of any flow in the graph is upper bounded by the sum of the capacities for any $u - v$ cut (Ahuja et al., 2013, property 6.1). In the remainder of the section whenever we mention a cut we mean a $u - v$ cut unless specified otherwise. If we consider some cut $\overline{\chi}_p \subseteq \overline{\mathcal{A}}_p$ in the graph $\overline{\mathcal{G}}_p$ then the sum of the capacities in the cut must be greater than or equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ for the solution to be feasible. So given the solution $(\overline{x}, \overline{y}, \overline{z})$ we solve the maximum flow problem by the labeling algorithm described by Ahuja et al. (2013, chapter 6). This algorithm also finds a minimum cut $\overline{\chi}_p \subseteq \overline{\mathcal{A}}_p$ and if the value of the maximum flow (capacity of the minimum cut) is less than $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ we add the following cut to the master problem:

$$\sum_{(u,c)\in\overline{\chi}_p} x_{c,p} + \sum_{(c,r)\in\overline{\chi}_p} z_{c,r} + \left|(r,v)\in\overline{\chi}_p\right| \geq \sum_{(u,c)\in\overline{\mathcal{A}}_p} x_{c,p} \tag{5.42}$$

For ease of notation in (5.42) we write $(u,c)\in\mathcal{A}$ which means; for each $c\in\mathcal{C}$ where the arc (u,c) is in the set $\mathcal{A}$. Similarly, $(r,v)\in\mathcal{A}$ means for each room $r\in\mathcal{R}$ where the arc (r,v) is in the set $\mathcal{A}$ and $(c,r)\in\mathcal{A}$ is the set of all the arcs (c,r) for each $c\in\mathcal{C}$ and $r\in\mathcal{R}$ in the set $\mathcal{A}$. We use the notation $|\cdot|$ on these sets to denote the cardinalities. We use this notation throughout the remainder of the paper.

We can rewrite (5.42) into a simpler form:

$$\sum_{(u,c)\in\overline{\mathcal{A}}_p\setminus\left\{\overline{\chi}_p\right\}} x_{c,p} - \sum_{(c,r)\in\overline{\chi}_p} z_{c,r} \leq \left|(r,v)\in\overline{\chi}_p\right| \tag{5.43}$$

Whenever we get an integer solution $(\overline{x},\overline{y},\overline{z})$ we add the cut (5.43) for each period $p\in\mathcal{P}$ where it is violated.

5.3 Primal Heuristic

The solutions obtained are often infeasible during the Branch-and-Bound process, which means that it is hard for the MIP solver to achieve upper bounds used for pruning. Therefore, we have implemented a heuristic to regain feasibility whenever a cut is generated to help the solver. In the heuristic we exploit the property that the courses can be assigned to any room. Due to constraints (5.28) we know that given any solution $(\overline{x},\overline{y},\overline{z})$ the time-schedule part $\overline{x}$ of the solution is feasible, meaning that it is possible to create a room assignment based in the time-schedule to obtain a complete solution. The heuristic runs through every period and in each period the lectures that are scheduled in that period are each assigned to a distinct room. This will always be possible since no more courses are scheduled in any period than the number of rooms available. Based on this assignment we then calculate the values of the y and z variables.

For each period $p\in\mathcal{P}$ we find a minimum cut $\overline{\chi}_p$ in the graph $\overline{\mathcal{G}}_p$ as described in section 5.2.2.2. Then we calculate the value $v_p\left(\overline{\chi}_p\right)$:

$$v_p\left(\overline{\chi}_p\right) := \sum_{(u,c)\in\overline{\mathcal{A}}_p\setminus\left\{\overline{\chi}_p\right\}} \overline{x}_{c,p} - \sum_{(c,r)\in\overline{\chi}_p} \overline{z}_{c,r} - \left|(r,v)\in\overline{\chi}_p\right| \tag{5.44}$$

If $v_p\left(\overline{\chi}_p\right) > 0$ then we cannot schedule all lectures in p and we need to change the values of the z variables to obtain feasibility. We do this by ordering the periods according to the values of $v_p\left(\overline{\chi}_p\right)$ from the largest to the smallest. We then iterate over the periods in the given order and for each period $p\in\mathcal{P}$ we perform the following steps:

1. Let the graph $\overline{\mathcal{G}}_p$ be defined as in section 5.2.2.2 with the same capacities. For each course $c\in\mathcal{C}$ and room $r\in\mathcal{R}$ let the weight on the arc $(c,r)\in\overline{\mathcal{A}}_p$ be $W^{\mathbf{RC}}\left(S_c - C_r\right)^+$ and let the weights of the remaining arcs be zero.

2. If $v_p\left(\overline{\chi}_p\right)\leq 0$, then stop.

3. For every arc $(c, r) \in \overline{\chi}_p$, if $\overline{z}_{c,r}$ (the capacity) is zero then we *open the room*, i.e., we change the value to one.

4. We then find the minimum cost maximum flow $\overline{f}$ (minimum cut $\overline{\chi}_p$) and for every arc (c, r) were we changed the capacity in step 3, if there is no flow on the arc, i.e., if $\overline{f}_{c,p,r} = 0$, we change the capacity back to zero.

5. Given the new minimum cut $\overline{\chi}_p$, update the value $v_p\left(\overline{\chi}_p\right)$ and go back to step 2.

When we have processed all periods we set the values $\overline{y}$ as follows:

$$\overline{y}_{c,r} = \sum_{p \in \mathcal{P}} \overline{f}_{c,p,r} \quad c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{5.45}$$

We then set the $\overline{z}$ values as follows to ensure that the constraints (5.18) are fulfilled:

$$\overline{z}_{c,r} = \begin{cases} 1 & \text{if } \overline{y}_{c,r} > 0 \\ 0 & \text{otherwise} \end{cases} \quad c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{5.46}$$

In step 3 we solve a minimum cost maximum flow problem, which is to find a minimum-cost flow among all the maximum flows. The reason we are solving a minimum cost maximum flow problem instead of just a maximum flow problem is that we want to create a feasible solution without increasing the penalty of the **RStab** constraint too much. The algorithm we use is a modified version of the algorithm described by Ahuja et al. (2013, section 9.7) which finds the minimum cost flow given the demands and supplies on the nodes. We have a supply on the source node and a demand on the sink node both equal to $\sum_{c \in \mathcal{C}} \overline{x}_p$ and on the rest of the nodes, the demands and supplies are zero. The modification to the algorithm is the stopping criterion. In its pure form the algorithm assumes that there it is possible to meet all demands of the nodes. Since we do not always have a feasible flow, we add the stopping criterion that if no path exists from the source node to the sink node in the residual graph, then the algorithm must stop. So the modified algorithm finds the minimum cost flow among all the maximum flows that exists. We can find the minimum cut from this flow in the same way as we did in section 5.2.2.2.

We need to show that the algorithm is finite and returns a feasible solution. If the algorithm assigns all lectures to rooms in each period, then the overall solution is feasible. So we need only to show that the algorithm is finite and returns a feasible solution for a single period since there is a finite number of periods. For period $p \in \mathcal{P}$, if all the arcs $(c, r) \in \overline{\mathcal{A}}_p$ have a capacity of one then all lectures scheduled in the period can be assigned to a room. So if the period is infeasible then some of the arcs $(c, r) \in \overline{\mathcal{A}}_p$ must have a capacity of zero. As long as at least one of the lectures cannot be assigned a room, i.e., as long as $v_p\left(\overline{\chi}_p\right) > 0$, the algorithm changes at least one value $\overline{z}_{c,r}$ from zero to one for some arc (c, r) in each iteration, assuming step 3 is correct. The correctness of step 3 is given by proposition 5.2. Since there is a finite number of these arcs, then the algorithm must also be finite. Furthermore, since the algorithm does not stop as long as $v_p\left(\overline{\chi}_p\right) > 0$ then the solution will also be feasible, for the period $p \in \mathcal{P}$, since all capacities on the arcs $(c, r) \in \overline{\mathcal{A}}_p$ are set to one in the worst case.

Proposition 5.2. *For a solution* $(\overline{x}, \overline{y}, \overline{z})$*, a period* $p \in \mathcal{P}$ *and a minimum cut* $\overline{\chi}_p$*, if* $v_p\left(\overline{\chi}_p\right) > 0$ *then there must exist a course* $c \in \mathcal{C}$ *and room* $r \in \mathcal{R}$ *where the arc* (c, r) *is in the cut* $\overline{\chi}_p$ *and the capacity is zero, i.e.,* $\overline{z}_{c,r} = 0$

To prove proposition 5.2 we first prove that in the minimum cut $\overline{\chi}_p$, at least one of the arcs $(c, r) \in \overline{\mathcal{A}}_p$ must be in the cut. Afterwards, we prove that the arcs $(c, r) \in \overline{\chi}_p$ cannot all have a capacity of one.

Proof. At least one arc $(c, r) \in \overline{\mathcal{A}}_p$ *is in the cut* $\overline{\chi}_p$. We begin the proof by showing that not all the arcs $(u, c) \in \overline{\mathcal{A}}_p$ can be in the minimum cut $\overline{\chi}_p$ nor can all the arcs $(r, v) \in \overline{A}_p$. In Figure 5.3 we give an illustration of the graph $\mathcal{G}_p$ and two cuts; $\overline{\chi}_p^1$ and $\overline{\chi}_p^2$.

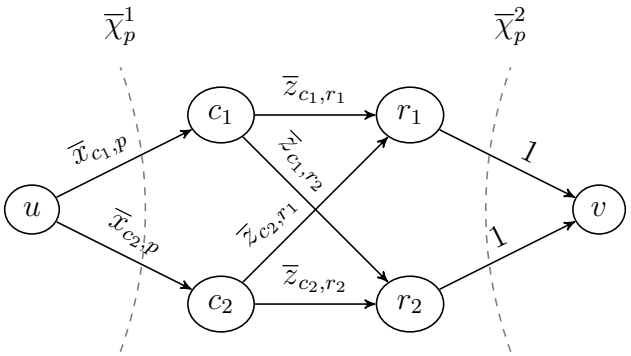


Figure 5.3: Illustration of the two cuts; cut $\overline{\chi}_p^1$ containing all arcs leaving the source node and $\overline{\chi}_p^2$ containing all arcs entering the sink node.

The cut $\overline{\chi}_p^1$ contains all the arcs $(u, c) \in \overline{\mathcal{A}}_p$ and the cut $\overline{\chi}_p^2$ contains all the arcs $(r, v) \in \overline{\mathcal{A}}_p$. Since we know $v_p\left(\overline{\chi}_p\right) < 0$ for the minimum cut $\overline{\chi}_p$ then neither $\overline{\chi}_p^1$ nor $\overline{\chi}_p^2$ can be a minimum cut. The reason that $\overline{\chi}_p^1$ cannot be a minimum cut is that the total capacity of the minimum cut $\overline{\chi}_p$ is strictly less than $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$, however, since the cut $\overline{\chi}_p^1$ contains all the arcs $(u, c) \in \overline{\mathcal{A}}_p$ then the total capacity of $\overline{\chi}_p^1$ must be greater than or equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$. The reason that $\overline{\chi}_p^2$ cannot be a minimum cut is that $\sum_{c \in \mathcal{C}} \overline{x}_{c,p} \leq |\mathcal{R}|$ due to constraint (5.28) thus the total capacity of the minimum cut $\overline{\chi}_p$ is strictly less than $|\mathcal{R}|$ since not all lectures are assigned to a room, however, since the cut $\overline{\chi}_p^2$ contains all the arcs $(r, v) \in \overline{\mathcal{A}}_p$ then the total capacity of $\overline{\chi}_p^2$ must be greater than or equal to $|\mathcal{R}|$.

As we cannot have all arcs $(u, c) \in \mathcal{A}_p$ in the minimum cut then there must exist some course $c \in \mathcal{C}$ where the node (c) is on the same side of the cut as the source node (u). Similarly, as we cannot have all arcs $(r, v) \in \mathcal{A}_p$ in the minimum cut then there must exist some room $r \in \mathcal{R}$ where the node (r) is on the same side of the cut as the sink node (v). Hence, the arc (c, r) must be in the cut which concludes our proof. □

We have shown that at least one of the arcs $(c, r) \in \overline{\mathcal{A}}_p$ must be in the minimum cut. One side effect of our proof is that since not all of the arcs $(u, c) \in \overline{\mathcal{A}}_p$ can be in the cut then we have the following:

$$\left|(u, c) \in \overline{\chi}_p\right| \leq |\mathcal{C}| - 1 \tag{5.47}$$

Similarly, since not all the arcs $(r, v) \in \overline{\mathcal{A}}_p$ can be in the minimum cut then we have the following:

$$\left|(r, v) \in \overline{\chi}_p\right| \leq |\mathcal{R}| - 1 \tag{5.48}$$

Before the next proof, we make the assumption that $|\mathcal{R}| \geq 2$. It is fair to make this assumption since if $|\mathcal{R}| = 0$ the problem is infeasible and if $|\mathcal{R}| = 1$ then at most one course is scheduled in each period and can be assigned to whichever room it has selected.

Proof. $\overline{z}_{c,r} = 0$ *for at least one arc* $(c, r) \in \overline{\chi}_p$. We showed in the previous proof that there is at least one arc (c, r) in the minimum cut for some course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$. Now we show for at least one of the arcs $(c, r) \in \overline{\chi}_p$ the capacity is zero. To prove this, we assume that every arc $(c, r) \in \overline{\chi}_p$ has a capacity of one and show that this leads to a contradiction.

Since we consider an infeasible solution then due to constraint (5.43) the following must hold:

$$\sum_{(c,r)\in\overline{\chi}_p} \overline{z}_{c,r} + \left|(r, v) \in \overline{\chi}_p\right| < \sum_{(u,c)\in\overline{\mathcal{A}}_p\backslash\overline{\chi}_p} \overline{x}_{c,p} \tag{5.49}$$

All capacities in the graph are either zero or one and so we can calculate an upper bound of the right-hand side of (5.49) as follows:

$$\sum_{(u,c)\in\overline{\mathcal{A}}_p\backslash\overline{\chi}_p} \overline{x}_{c,p} \leq \left|(u, c) \in \overline{\mathcal{A}}_p\backslash\overline{\chi}_p\right| = |\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right| \tag{5.50}$$

If we insert (5.50) into (5.49) together with our assumption that the capacities of all the arcs $(c, r) \in \overline{\chi}_p$ are one then we get the following:

$$\left|(c, r) \in \overline{\chi}_p\right| + \left|(r, v) \in \overline{\chi}_p\right| < |\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right| \tag{5.51}$$

Consider an arc (c, r) for some course $c \in \mathcal{C}$ and some room $r \in \mathcal{R}$. For this arc to be in the cut, the node (c) must be on the same side of the cut as the source node and the node (r) must be on the same side as the sink node. The node (c) is on the source side if and only if the arc (u, c) is not in the cut. Similarly, the node (r) is on the sink side if and only if the arc (r, v) is not in the cut. So we can express $\left|(c, r) \in \overline{\chi}_p\right|$ as the product of the arcs (u, c) and (r, v) that are not in the cut:

$$\left|(c, r) \in \overline{\chi}_p\right| = \left|(u, c) \in \overline{\mathcal{A}}_p\backslash\overline{\chi}_p\right| \left|(r, v) \in \overline{\mathcal{A}}_p\backslash\overline{\chi}_p\right| \tag{5.52}$$

We can rewrite (5.52) by noting that the number of arcs leaving the source that are not in the cut is equal to the number of arcs leaving the source minus the arcs in the cut, and similarly for the arcs entering the sink:

$$\left|(c, r) \in \overline{\chi}_p\right| = \left(|\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right|\right) \left(|\mathcal{R}| - \left|(r, v) \in \overline{\chi}_p\right|\right) \tag{5.53}$$

Expanding the expression gives us the following:

$$\begin{aligned}\left|(c, r) \in \overline{\chi}_p\right| =& |\mathcal{C}||\mathcal{R}| + \left|(u, c) \in \overline{\chi}_p\right| \left|(r, v) \in \overline{\chi}_p\right| \\ & - |\mathcal{R}| \left|(u, c) \in \overline{\chi}_p\right| - |\mathcal{C}| \left|(r, v) \in \overline{\chi}_p\right|\end{aligned} \tag{5.54}$$

Substituting (5.54) into (5.51) gives the following:

$$\begin{aligned} |\mathcal{C}||\mathcal{R}| + \left|(r,v) \in \overline{\chi}_p\right| < |\mathcal{C}| - \left|(u,c) \in \overline{\chi}_p\right| \left|(r,v) \in \overline{\chi}_p\right| \\ + |\mathcal{R}| \left|(u,c) \in \overline{\chi}_p\right| + |\mathcal{C}| \left|(r,v) \in \overline{\chi}_p\right| - \left|(u,c) \in \overline{\chi}_p\right| \end{aligned} \tag{5.55}$$

Next we rearrange the terms:

$$\begin{aligned} |\mathcal{C}|\left(|\mathcal{R}|-1\right) < \left(|\mathcal{C}|-1\right)\left(|\mathcal{R}|-1\right) \\ - \left(\left(|\mathcal{R}|-1\right) - \left|(r,v) \in \overline{\chi}_p\right|\right)\left(\left(|\mathcal{C}|-1\right) - \left|(u,c) \in \overline{\chi}_p\right|\right) \end{aligned} \tag{5.56}$$

An upper bound can be found for the right-hand side of (5.56) by finding a lower bound for the expression:

$$\left(\left(|\mathcal{R}|-1\right) - \left|(r,v) \in \overline{\chi}_p\right|\right)\left(\left(|\mathcal{C}|-1\right) - \left|(u,c) \in \overline{\chi}_p\right|\right) \tag{5.57}$$

Since we know that $\left|(r,v) \in \overline{\chi}_p\right| \leq |\mathcal{R}| - 1$ then $\left(\left(|\mathcal{R}|-1\right) - \left|(r,v) \in \overline{\chi}_p\right|\right) \geq 0$ must hold. Similarly $\left(\left(|\mathcal{C}|-1\right) - \left|(u,c) \in \overline{\chi}_p\right|\right) \geq 0$ must also hold which means that the minimum value that (5.57) can take must be zero. So the right-hand side of (5.56) must be less than or equal to $\left(|\mathcal{C}|-1\right)\left(|\mathcal{R}|-1\right)$ and since we assume $|\mathcal{R}| \geq 2$ we can divide with $|\mathcal{R}| - 1$ and we get the following:

$$|\mathcal{C}| < |\mathcal{C}| - 1 \tag{5.58}$$

As (5.58) is a contradiction then we have concluded our proof and at least one of the arcs $(c,r) \in \overline{\chi}_p$ must have a capacity of zero. □

We have now proven that our algorithm is finite and returns a feasible solution. We run this heuristic whenever we retrieve an integer solution where a cut is generated.

5.4 Computational Experiments

In this section, we report our computational experiments. We conducted all our tests on a 3.40GHz Intel® Core™ processor running Windows 10 with 8GB memory RAM. We used Gurobi 6.5.2 provided by Gurobi Optimization Inc. (2015) both for the master problem and also for the subproblem described in section 5.2.2.1. We set all parameters in Gurobi to their default except, *Presolve* , *Threads* and *LazyConstraints*. We set Presolve to 2 (the most aggressive level), *Threads* to 1 as this was the maximum allowed threads for the ITC2007 competition and *LazyConstraints* to 1 to be able to add cuts on integer solutions.

We tested our approach on four datasets; TEST, COMP, DDS and ERLANGEN. The TEST dataset consists of four data instances, test1 – test4, proposed by Di Gaspero and Schaerf (2003). The COMP dataset contains 21 data instances, comp01 – comp21, described in Di Gaspero et al. (2007) mainly taken from the University of Udine. The DDS dataset contains seven data instances, DDS1 – DDS7, taken from other Italian universities. The ERLANGEN dataset consists of six instances, erlangen2012_1, erlangen2012_2, erlangen2013_1,

erlangen2013_1, erlangen2013_2 and erlangen2014_1, provided by Dr.-Ing. Moritz Mühlenthaler. All the datasets can be retrieved from http://tabu.diegm.uniud.it/ctt/index.php. For the ITC2007 competition, a benchmarking tool was provided, which can be obtained from http://www.cs.qub.ac.uk/itc2007. The tool calculates the time limit for the competition on the machine the tool is executed on. This time limit is referred to as one CPU unit, which in our case is 208 seconds. We have written all our implementations (including the flow algorithms) in C#.

The flow diagram in Figure 5.4 is an illustration of our implementation of Benders' Decomposition.

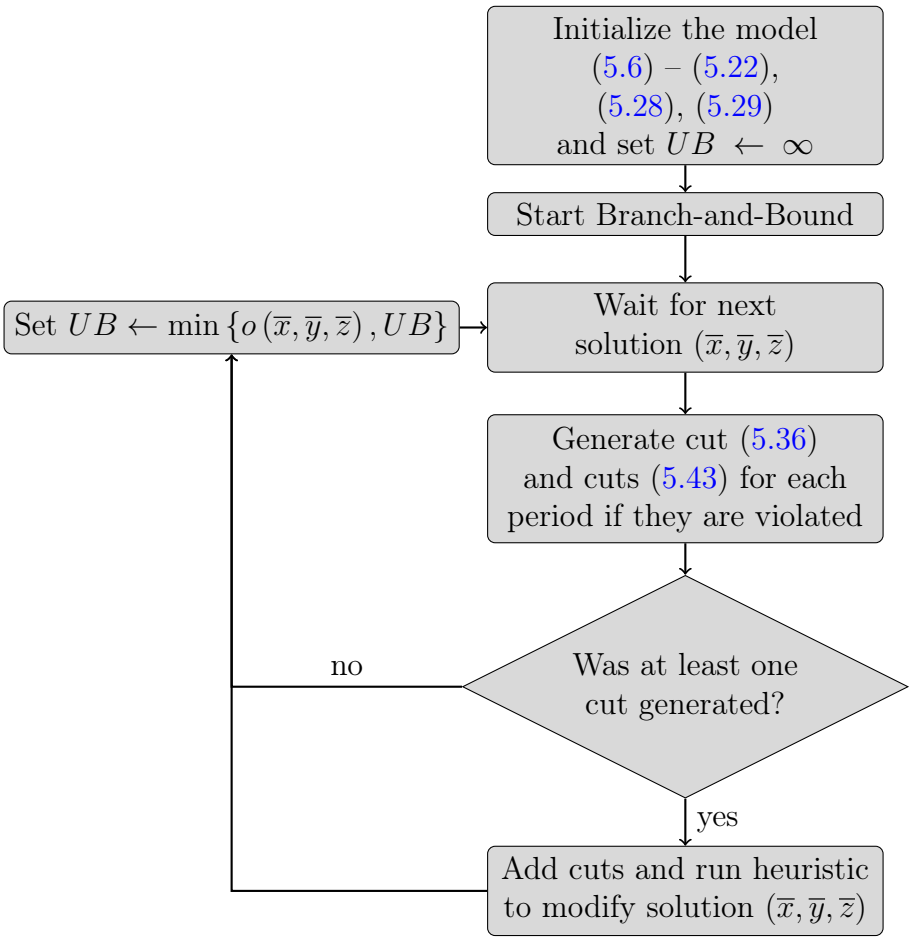


Figure 5.4: Flow diagram of the implemention of Benders' decomposition. The algorithm continues until a stopping criterion is met, e.g., when a time limit is reached or the best solution is proven optimal.

At first the model is initialized as (5.6) – (5.22), (5.28), (5.29) and we set the best known solution value to be infinity. We then start Gurobis Branch-and-Bound solver and in a callback function we retrieve every integer solution that is obtained. Whenever we receive an integer solution we generate the cut (5.36) and cuts (5.43) for each period if the solution violates them. If we generate at least one cut, then we run the heuristic described in section 5.3 to repair the solution. If the repaired solution is better than our current best-known solution, then we

reinsert the solution into Gurobi. The reason that we reinsert the solution is that Gurobi can use the solution to prune nodes in the Branch-and-Bound tree and the heuristics can use the solution to search for improvements.

In the following we compare our implementation with results obtained in the literature by other MIP-based methods; Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011) and Cacchiani et al. (2013). In section 5.4.1 we first compare the lower bounds obtained by our implementation with Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011) and Cacchiani et al. (2013) on the first fourteen data instances of COMP. Afterwards, we compare the lower bounds obtained on all 21 data instances in COMP as well as all instances from the sets DDS and TEST with Cacchiani et al. (2013) as they have reported lower bounds for all these sets. In section 5.4.2 we compare both lower and upper bounds obtained by Lach and Lübbecke (2012) and Burke et al. (2010) as these methods provide not only lower bounds but also upper bounds. In section 5.4.3 we compare our implementation to the original model without Benders' decomposition, i.e., model (5.6) – (5.26) where the integrality requirements (5.26) are relaxed such that the f variables are non-negative continuous variables.

Before these comparisons we compare the implementation of Benders' decomposition with the results from Bagger et al. (2016) when given a time limit of forty CPU units in Table 5.1; the minimum cost flow formulation (Min) and the multi-commodity flow formulation (Mult). If an approach obtains a lower bound for an instance which is at least as good as the other approaches for the same instance, then we denote this using a bold font. If the lower bound is better than all the other approaches, then we mark it with an underline. In the bottom of the tables, we report the number of times that each approach obtains a lower bound which is at least as good as the others (the line *Best*) and the number of times a lower bound which is better than all the other approaches (the line following *Best*). Furthermore, we compute the ranking of the approaches on each instance. For each instance, the approach that obtains the best bound is given rank 1, the second best is given rank 2 and so on. If multiple approaches are tied then they receive the average rank of the respective ranks, e.g., if two approaches are tied for rank 3 and 4 then they both receive rank 3.5 as this is the average. In the last line of the tables, we report the average of the ranks of each approach. The approach that obtains the best rank is marked with a bold font. We use this notation throughout all the tables.

In Table 5.1 we see that the implementation of Benders' decomposition obtains a lower bound which is at least as good as the other two in 27 instances and a bound which is better in 11 out of 32 instances. Referring to Bonutti et al. (2016) we observe that for two of the instances (marked with an * in the table), comp12 and test4, the lower bounds we obtain, 147 and 49, is an improvement of the currently best-known bounds of 142 and 46 respectively. The upper bounds that we obtain in Benders' decomposition is never better than the Min nor the Mult approaches and has an upper bound which is equal to at least one of the other on six out of 32 instances. Thus, the current focus of the heuristic on feasibility is not enough to improve the upper bounds. Future work could include the possibility of including improvement techniques after the heuristic has regained feasibility.

Table 5.1: Comparison of the lower bounds obtained on COMP, DDS and TEST for Min (minimum cost flow formulation from Bagger et al. (2016)), Mult (multi-commodity flow formulation from Bagger et al. (2016)) and Benders (our implementation in this article) with a time limit of forty CPU units.

Instance	Min		Mult		Benders	
	LB	UB	LB	UB	LB	UB
comp01	**5**	**5**	**5**	**5**	**5**	**5**
comp02	8	**45**	8	59	**9**	112
comp03	38	123	37	**99**	**42**	107
comp04	**35**	**35**	**35**	**35**	**35**	76
comp05	**186**	**355**	181	377	168	466
comp06	16	**92**	16	**92**	**20**	139
comp07	0	**179**	**6**	-	**6**	183
comp08	**37**	**37**	**37**	41	**37**	99
comp09	74	105	73	**103**	**82**	113
comp10	**4**	**18**	**4**	68	**4**	72
comp11	**0**	**0**	**0**	**0**	**0**	**0**
comp12	142	**423**	140	500	**147***	464
comp13	56	66	**59**	**59**	57	152
comp14	44	**55**	43	74	**48**	93
comp15	38	123	37	**99**	**42**	107
comp16	13	61	11	**42**	**16**	76
comp17	43	123	44	**109**	**46**	132
comp18	**36**	**78**	30	108	35	88
comp19	**56**	**57**	55	**57**	54	92
comp20	**0**	**50**	**0**	96	**0**	340
comp21	56	156	57	**133**	**62**	138
DDS1	**46**	**70**	44	76	45	214
DDS2	**0**	**0**	**0**	**0**	**0**	**0**
DDS3	**0**	**0**	**0**	**0**	**0**	**0**
DDS4	**15**	**17**	**15**	12079	**15**	1192
DDS5	**0**	**0**	**0**	**0**	**0**	**0**
DDS6	**0**	**2**	**0**	27	**0**	123
DDS7	**0**	**0**	**0**	**0**	**0**	**0**
test1	**224**	**224**	**224**	**224**	**224**	314
test2	**16**	**19**	**16**	**19**	**16**	104
test3	**59**	**75**	**59**	**75**	**59**	96
test4	44	**91**	43	107	**49***	101
Best	**19**	**25**	**17**	**19**	**27**	**6**
	4	**13**	**1**	**7**	**11**	
Rank	2.03	**1.59**	2.27	1.84	**1.70**	2.56

5.4.1 Lower Bounds

In this section we first compare the lower bounds obtained by our approach with Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011) and Cacchiani et al. (2013). Following the literature we compare the results using three time limits; one CPU unit, ten CPU units and forty CPU units. We start by comparing our results with Cacchiani et al. (2013) as they are the only ones to report lower bounds for the entire COMP dataset as well as for DDS and TEST for forty CPU units. In Table 5.2 we compare their results with our approach where we use the same notation as in Table 5.1.

In Table 5.2 we see that our approach obtains a bound which is at least as good as the bound obtained by Cacchiani et al. (2013) on 23 of the 32 instances and a better bound on eight of the 32 instances.

Next, we compare the lower bounds obtained by our approach with Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011) and Cacchiani et al. (2013) on the first fourteen data instances of COMP. The reason for only comparing the first fourteen instances is that these were the only ones that were available for Lach and Lübbecke (2012) and Burke et al. (2010). In the tables 5.3, 5.4 and 5.5 we report the lower bounds obtained for each approach given a time limit of one, ten and forty CPU units respectively. We use the same notation as in Table 5.1.

In Table 5.3 – 5.5 we see that our approach obtains a lower bound which is at least as good as the other approaches in ten instances for one and ten CPU units and in eleven out of the fourteen instances for forty CPU units. Benders' decomposition obtain a better bound in four, three and three out of the fourteen instances for one, ten and forty CPU units respectively. Considering the average rank obtained we see that our approach obtains a lower average rank than all the other approaches on the first fourteen data instances of COMP for all three time limits.

5.4.2 Upper Bounding Methods

In this section, we compare both the upper bounds that we obtain on the first fourteen data instances from COMP with Lach and Lübbecke (2012) and Burke et al. (2010) as these are the only methods besides ours that obtain both lower and upper bounds. We report the results in the tables 5.6 – 5.8 where we use the same notation as in Table 5.1.

Regarding the upper bounds we see that in Table 5.6 we obtain upper bounds which are better on six out of the fourteen instances and the two other approaches both obtain a better upper bound on four of the instances each. For both ten (Table 5.7) and forty (Table 5.7) CPU units our approach obtains a better upper bound for three of the instances, Lach and Lübbecke (2012) obtains a better upper bound on ten and 8 instances respectively and Burke et al. (2010) obtains a better upper bound on one and three instances respectively.

Table 5.2: Comparison of the lower bounds obtained on COMP, DDS and TEST for CCRT13 (Cacchiani et al., 2013) and Benders (our implementation) with a time limit of forty CPU units.

Instance	CCRT13	Benders
comp01	**5**	**5**
comp02	**16**	9
comp03	**52**	42
comp04	**35**	**35**
comp05	166	**168**
comp06	11	**20**
comp07	**6**	**6**
comp08	**37**	**37**
comp09	**92**	82
comp10	2	**4**
comp11	**0**	**0**
comp12	100	**147**
comp13	**57**	**57**
comp14	**48**	**48**
comp15	**52**	42
comp16	13	**16**
comp17	**48**	46
comp18	**52**	35
comp19	48	**54**
comp20	**4**	0
comp21	**68**	62
DDS1	40	**45**
DDS2	**0**	**0**
DDS3	**0**	**0**
DDS4	**17**	15
DDS5	**0**	**0**
DDS6	**0**	**0**
DDS7	**0**	**0**
test1	**224**	**224**
test2	**16**	**16**
test3	**59**	**59**
test4	46	**49**
Best	**24**	**23**
	9	**8**

Table 5.3: Comparison of the lower bounds obtained on the first fourteen data instances of COMP for the different approaches with a time limit of one CPU unit; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Benders (our implementation).

Instance	LL12	BMPR10	HB11	CCRT13	Benders
comp01	4	0	4	**5**	**5**
comp02	0	0	**10**	0	7
comp03	0	25	26	24	**37**
comp04	22	**35**	**35**	**35**	**35**
comp05	92	119	19	6	**145**
comp06	7	13	12	0	**17**
comp07	0	**6**	5	0	**6**
comp08	30	**37**	**37**	**37**	**37**
comp09	37	68	39	**92**	78
comp10	2	3	**4**	0	**4**
comp11	**0**	**0**	**0**	**0**	**0**
comp12	29	**101**	43	0	91
comp13	33	52	46	**57**	56
comp14	40	41	41	32	**43**
Best	**1**	**5** **1**	**5** **1**	**6** **2**	**10** **4**
Rank	4.21	2.71	2.82	3.50	**1.75**

Table 5.4: Comparison of the lower bounds obtained on the first fourteen data instances of COMP for the different approaches with a time limit of ten CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Benders (our implementation).

Instance	LL12	BMPR10	HB11	CCRT13	Benders
comp01	4	4	4	**5**	**5**
comp02	8	0	12	**16**	8
comp03	0	33	34	**52**	41
comp04	28	**35**	**35**	**35**	**35**
comp05	25	111	69	6	**167**
comp06	10	15	12	11	**19**
comp07	2	**6**	**6**	**6**	**6**
comp08	34	**37**	**37**	**37**	**37**
comp09	41	65	67	**92**	81
comp10	**4**	**4**	**4**	2	**4**
comp11	**0**	**0**	**0**	**0**	**0**
comp12	32	95	78	0	**144**
comp13	39	52	53	**57**	**57**
comp14	41	42	43	**48**	47
Best	**2**	**5**	**5**	**10** **4**	**10** **3**
Rank	4.36	3.14	2.86	2.61	**2.04**

Table 5.5: Comparison of the lower bounds obtained on the first fourteen data instances of COMP for the different approaches with a time limit of forty CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Benders (our implementation).

Instance	LL12	BMPR10	HB11	CCRT13	Benders
comp01	4	**5**	4	**5**	**5**
comp02	11	1	12	**16**	9
comp03	25	33	36	**52**	42
comp04	28	**35**	**35**	**35**	**35**
comp05	108	114	80	166	**168**
comp06	10	16	16	11	**20**
comp07	**6**	**6**	**6**	**6**	**6**
comp08	**37**	**37**	**37**	**37**	**37**
comp09	46	66	67	**92**	82
comp10	**4**	**4**	**4**	2	**4**
comp11	**0**	**0**	**0**	**0**	**0**
comp12	53	95	84	100	**147**
comp13	41	54	55	**57**	**57**
comp14	46	42	43	**48**	**48**
Best	**4**	**6**	**5**	**10** **3**	**11** **3**
Rank	4.00	3.32	3.21	2.32	**2.14**

Table 5.6: Comparison of the upper bounds (UB) obtained on the first fourteen data instances of COMP for the different approaches with a time limit of one CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010) and Benders (our implementation).

Instance	LL12	BMPR10	Benders
comp01	**12**	168	14
comp02	239	**114**	118
comp03	194	**158**	179
comp04	**44**	153	108
comp05	965	1447	**604**
comp06	395	277	**227**
comp07	525	-	**260**
comp08	**78**	173	106
comp09	115	**112**	142
comp10	235	**70**	101
comp11	7	288	**0**
comp12	1122	-	**874**
comp13	**98**	556	229
comp14	113	123	**103**
Best	**4**	**4**	**6**
	4	**4**	**6**
Rank	2.00	2.36	**1.64**

Table 5.7: Comparison of the upper bounds (UB) obtained on the first fourteen data instances of COMP for the different approaches with a time limit of ten CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010) and Benders (our implementation).

Instance	LL12	BMPR10	Benders
comp01	12	10	**5**
comp02	**93**	101	112
comp03	**86**	144	122
comp04	41	**36**	76
comp05	**468**	649	581
comp06	**79**	317	139
comp07	**28**	857	191
comp08	**48**	53	99
comp09	**106**	115	113
comp10	**44**	49	72
comp11	7	12	**0**
comp12	657	889	**472**
comp13	**67**	92	222
comp14	**54**	72	93
Best	**10**	**1**	**3**
	10	**1**	**3**
Rank	**1.36**	2.43	2.21

Table 5.8: Comparison of the upper bounds (UB) obtained on the first fourteen data instances of COMP for the different approaches with a time limit of forty CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010) and Benders (our implementation).

Instance	LL12	BMPR10	Benders
comp01	12	9	**5**
comp02	**46**	63	112
comp03	**66**	123	107
comp04	38	**36**	76
comp05	**368**	629	466
comp06	51	**46**	139
comp07	**25**	45	183
comp08	44	**41**	99
comp09	**99**	105	113
comp10	**16**	23	72
comp11	7	12	**0**
comp12	548	785	**464**
comp13	**66**	67	152
comp14	**53**	55	93
Best	**8**	**3**	**3**
	8	**3**	**3**
Rank	**1.50**	2.07	2.43

5.4.3 Large Datasets (Erlangen)

In this section, we test our model on the ERLANGEN dataset. The data instances in this set are larger than for the other datasets we have tested and for some of the instances even solving the root note LP takes longer time than the forty CPU units we tested on. Therefore we have tested them using 100 CPU units, and we have changed the parameter setting to use the non-homogeneous barrier method as the LP solver (Method=2, NodeMethod=2 and BarHomogeneous=0) and with the number of threads to be equal to the number of processors (Threads=4) as this accelerates the solution time of the LP. We compare our implementation to the original model (the model that we have based our decomposition on) to see the benefits of the decomposition for such large datasets.

In Table 5.9 we report the lower and upper bounds obtained after running the MIP solver for the given time limit (100 CPU) for both the original model and the decomposition. In the last two columns we report how much the values are improved by Benders. In the column *LB* we report the increase of the lower bound in percent which is calculated as $(LB_{Benders} - LB_{Mult})/LB_{Mult}$ where LB_{Mult} and $LB_{Benders}$ are the lower bounds obtained by the original and the decomposed model respectively. Similarly, in column *UB* we report the decrease of the upper bound.

Table 5.9: Comparison of the lower bound (LB) and upper bound (UB) obtained for the original model (Mult) and our implementation (Benders) on the ERLANGEN dataset when running the Branch-and-Bound solver Gurobi. The last two columns (Impr.) report how much the bound is improved by Benders in percentages.

	MULT		Benders		Impr.	
Instance	LB	UB	LB	UB	LB	UB
2011_2	2105	10390	**2385**	**10127**	13%	3%
2012_1	1235	21178	**1393**	**18334**	13%	13%
2012_2	1556	-	**1766**	**21722**	13%	-
2013_1	1259	172474	**1417**	**28529**	13%	83%
2013_2	1086	-	**1274**	**19808**	17%	-
2014_1	834	30985	**975**	**18851**	17%	39%
Avg.					14%	35%

We see in Table 5.9 that the decomposition approach obtains better bounds for all six instances, where for two of the instances the original model does not obtain any solution. Our decomposition gives an improvement of 14% and 35% on average for the lower and upper bounds respectively. However, it should be noted that the average improvement on the upper bounds is calculated on only four of the six insrances

In Table 5.10 we report the performance for the root LP, i.e., the results of solving the LP relaxation in the root node before the MIP solver starts to add cuts and branch. For both the original model and the decomposition we report the time spent by the solver in seconds (Time) and the objective value of the LP (Obj). In the last two columns we report how much the values are improved be Benders. In the column, *Time* we report the speed-up factor which we calculate as the time that the solver spent on solving the root node LP in the original model divided by the time spent in the decomposed model. In the column *Obj* we report the increase

of the objective value in permille which is calculated as $(Obj_{Benders} - Obj_{Mult})/Obj_{Mult}$ where Obj_{Mult} and $Obj_{Benders}$ are the objective values of the root node LP in the original and the decomposed model respectively.

Table 5.10: Comparison of the time (Time) it takes to solve the root node LP and the objective value (Obj) obtained for the original model (Mult) and our implementation (Benders) on the ERLANGEN dataset. The last two columns (Impr.) report how much the values are improved by Benders. The improvement of the time is reported as the speed-up factor and the improvement of the objective value is reported in permille. The last line (Avg.) report the average improvements.

	Mult		Benders		Impr.	
Instance	Time	Obj	Time	Obj	Time	Obj
2011_2	5836.7	1985.7	**86.6**	**1994.8**	×57	5‰
2012_1	2518.9	1015.8	**117.7**	**1019.8**	×21	4‰
2012_2	6306.5	1347.9	**162.3**	**1347.9**	×39	0‰
2013_1	3083.1	1030.9	**153.5**	**1032.4**	×20	1‰
2013_2	5634.6	959.9	**199.9**	**967.7**	×28	8‰
2014_1	2740.6	734.9	**224.8**	**737.6**	×12	4‰
Avg.					×31	4‰

In Table 5.10 we see that our decomposition speeds up the solution time for the root node LP by more than thirty times on average. This speed-up makes sense as the majority of the variables in the model are projected out in the decomposition. What may come as a surprise is that the objective value for five of the instances is larger in the decomposition than in the original model. The reason for this improvement is that the MIP solver manages to heuristically generate some integer solutions for the decomposition before it starts to solve the LP. These integer solutions are infeasible and some of the cuts we generate are the cuts (5.43) which are based on more information than what is available for the original model, i.e., these cuts are based on model (5.39) – (5.41). In our opinion, these results illustrate that Benders' decomposition is a powerful tool for this problem, especially for massive datasets.

5.5 Conclusion

In this article, we proposed a Benders' decomposition on a Mixed Integer Programming (MIP) model for the Curriculum-based Course Timetabling problem. We also described a heuristic to repair the solutions that are cut away by the Benders' feasibility cuts. Regarding lower bounds, our approach obtained better results on average compared to other MIP based approaches for the first fourteen instances from the second international timetabling competition. We compared our approach on a larger set of data (a total of 32 instances) with the state-of-the-art algorithm. We obtained a lower bound which was at least good on 23 of the 32 instances and we obtained a better bound on eight instances. Furthermore we improved the currently best known lower bounds for two out of the 32 instances. Benders' decomposition did not provide better upper bounds than the non-decomposed model, thus, the focus on feasibility in the heuristic is not enough to improve the upper bounds. Lastly, we compared Benders'

decomposition to the MIP model that the decomposition originates from on six additional data instances that are much larger than the other 32 instances. The results showed the benefit of implementing Benders' decomposition as both the lower and upper bounds were better for all the instances compared to the original model. To our knowledge, no other studies have tested MIP based methods on these data instances, so we improved the best known lower bounds on all six instances.

Acknowledgments

The authors would like to thank Assistant Professor Evelien van der Hurk for her valuable feedback to improve the manuscript.

Funding: This work is part of an industrial PhD project funded by Innovation Fund Denmark (IFD). IFD has supported this work solely financially and has not taken part in any research related activities.

References

Ahuja, R., Magnanti, T., and Orlin, J. (2013). *Network Flows: Theory, Algorithms, and Applications*. Always learning. Pearson Education Limited. ISBN: 9781292042701. URL: https://books.google.ca/books?id=rFuLngEACAAJ.

Bagger, N.-C. F., Simon, K., Sørensen, M., and Stidsen, T. R. (2016). *Flow Formulations for Curriculum-based Course Timetabling*. Available on Optimization Online. URL: http://www.optimization-online.org/DB_HTML/2016/12/5786.html.

Benders, J. (1962). "Partitioning procedures for solving mixed-variables programming problems". English. In: *Numerische Mathematik* 4.1, pp. 238–252. ISSN: 0029-599X. DOI: 10.1007/BF01386316. URL: http://dx.doi.org/10.1007/BF01386316.

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). "An overview of curriculum-based course timetabling". In: *TOP*, pp. 1–37.

Bonutti, A., De Cesco, F., Di Gaspero, L., and Schaerf, A. (2012). "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results". In: *Annals of Operations Research* 194.1, pp. 59–70.

Bonutti, A., Gaspero, L. D., and Schaerf, A. (2016). *Curriculum-Based Course TimeTabling*. http://tabu.diegm.uniud.it/ctt/index.php[Retrieved 30/12-2016].

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2008). "Penalising Patterns in Timetables: Novel Integer Programming Formulations". In: *Operations Research Proceedings 2007*. Ed. by J. Kalcsics and S. Nickel. Vol. 2007. Operations Research Proceedings. 10.1007/978-3-540-77903-2_63. Springer Berlin Heidelberg, pp. 409–414. ISBN: 978-3-540-77903-2.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010). "A supernodal formulation of vertex colouring with applications in course timetabling". In: *Annals of Operations Research* 179.1, pp. 105–130.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). "A branch-and-cut procedure for the Udine Course Timetabling problem". In: *Annals of Operations Research* 194.1, pp. 71–87.

Cacchiani, V., Caprara, A., Roberti, R., and Toth, P. (2013). "A new lower bound for curriculum-based course timetabling". In: *Computers and Operation Research* 40.10, pp. 2466–2477. DOI: 10.1016/j.cor.2013.02.010.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). *The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)*. Tech. rep. School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom.

Di Gaspero, L. and Schaerf, A. (2003). "Multi-neighbourhood Local Search with Application to Course Timetabling". English. In: *Practice and Theory of Automated Timetabling IV*. Ed. by E. Burke and P. De Causmaecker. Vol. 2740. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 262–275. ISBN: 978-3-540-40699-0. DOI: 10.1007/978-3-540-45157-0_17. URL: http://dx.doi.org/10.1007/978-3-540-45157-0_17.

Fischetti, M. (2015). *The simpler the better: Thinning out MIP's by Occam's razor*. Presentation at CORS/INFORMS 2015 Joint International Meeting in Montréal. Slides retrieved December 16, 2016, from: http://www.dei.unipd.it/~fisch/papers/slides/2015%20CORS%20%5bFischetti%20-%20Occam%20Razor%5d.pdf.

Geoffrion, A. M. (1972). "Generalized benders decomposition". In: *Journal of optimization theory and applications* 10.4, pp. 237–260.

Gurobi Optimization Inc. (2015). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Hao, J. K. and Benlic, U. (2011). "Lower bounds for the ITC-2007 curriculum-based course timetabling problem". In: *European Journal of Operational Research* 212.3, pp. 464–472.

Lach, G. and Lübbecke, M. (2012). "Curriculum based course timetabling: new solutions to Udine benchmark instances". In: *Annals of Operations Research* 194, pp. 255–272. ISSN: 0254-5330.

Lach, G. and Lübbecke, M. E. (2008). "Optimal University Course Timetables and the Partial Transversal Polytope". In: *International Workshop on Experimental and Efficient Algorithms*. Springer, pp. 235–248.

Lübbecke, M. E. (2015). "Comments on: An overview of curriculum-based course timetabling". In: *TOP* 23.2, pp. 359–361.

Martin, R. K. (1999). *Large scale linear and integer optimization: a unified approach*. Kluwer Academic Publishers.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., and Burke, E. K. (2010). "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition". In: *INFORMS Journal on Computing* 22.1, pp. 120–130.

Part III

Lower Bounding Methods

6 Daily Course Pattern Formulation and Valid Inequalities for the Curriculum-based Course Timetabling Problem

Niels-Christian F. Bagger[a,b] · Guy Desaulniers[c,d] · Jacques Desrosiers[e]

[a]mOReTime research group, Management Science, Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 426B, DK-2800 Kgs. Lyngby, Denmark, http://www.moretime.man.dtu.dk/

[b]MaCom A/S, Vesterbrogade 48, 1., DK-1620 København V, Denmark

[c]GERAD – HEC Montréal 3000, Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[d]Polytechnique Montréal, Montréal, Canada H3C 3A7

[e]HEC Montréal, Montréal, Canada H3T 2A7

Status: Submitted to Journal of Scheduling

Abstract: In this paper, we propose an integer programming model for obtaining lower bounds for the curriculum-based course timetabling problem, in which weekly assignments for courses into rooms and periods are considered. The model is a pattern formulation where a pattern is an assignment of a course into a set of periods on one day. Different preprocessing techniques are implemented to reduce the number of variables and valid inequalities are derived and added to the model. The proposed model is tested on 21 real-world data instances. On 17 of these instances the best know solutions have been proven optimal and for the remaining four our model improves the lower bounds for three of them.

Keywords: Course Timetabling · Integer Programming · Preprocessing · Valid Inequalities

6.1 Introduction

The problem considered in this paper is the *Curriculum-based Course Timetabling Problem* (CCT) as described by Di Gaspero et al. (2007). The interest in this problem has increased as it was used as one of three problems for the *Second International Timetabling Competition* (ITC2007) in 2007 (Di Gaspero et al., 2007; McCollum et al., 2010). In connection to ITC2007, a website was developed (Bonutti et al., 2016) where it is possible for researchers to upload solutions and to report refined lower bounds. Twenty-one data sets were provided for the competition based on real world instances. Seven of these were hidden until after the competition. For the 21 data sets, most of them has been *closed*, meaning that the best solutions found have been proven optimal. However there are still four *open* data sets left and the aim of this paper is to reduce the gaps by improving the lower bounds.

6.1.1 Problem Description

A set of courses is given, each having a predetermined number of lectures to be scheduled. Each course is taught be exactly one lecturer. A set of curricula is also given, each consisting of a set of courses. A course can belong to several curricula. Furthermore a time horizon in terms of a number of days that all the lectures need to be scheduled in is given. Each day is divided into a number of uniform time slots and a day and time slot pair is referred to as a period. It is assumed that each lecture takes up exactly one period. Besides scheduling the lectures into periods, it is also needed to choose a room for the lecture to take place in. Finally some constraints are also given. These can either be *hard* or *soft*. Any feasible timetable should fulfil all the hard constraints, i.e., a timetable is considered feasible if and only if all the hard-constraints have no violations. The objective is then to find a feasible timetable while minimizing a weighted sum of the violation of the soft constraints.

The problem formulated by Di Gaspero et al. (2007) contains four hard constraints as listed below:

Availability (A): A course can have specific periods defined as *unavailable* periods. Every lecture scheduled in such a period is considered as one violation.

Conflicts (C): A pair of lectures is considered as conflicting if they belong to the same curriculum or are taught by the same lecturer. If a conflicting pair of lectures is scheduled in the same period, this is considered as one violation.

Lectures (L): Each course has a predetermined number of lectures to be scheduled in different periods. If a lecture is not scheduled or if two lectures of a course are scheduled in the same period, this is counted as one violation.

Room Occupancy (RO): For every room and every period, at most one lecture should be scheduled. For every room and period, every additional lecture is considered as one violation.

In our approach, we are only focusing on scheduling the courses into the periods. The only hard constraint involving the rooms is **RO** and this constraint can still be fulfilled since it is possible to put any course into any room. Therefore we just have to ensure that we do not

schedule more courses into a single period than the number of available rooms in the data instance. The problem also contains four soft constraints for which we minimize the number of violations. These are listed below:

Isolated Lectures (IL): It is desired to create compact schedules. A lecture is *isolated* if no other lectures are scheduled for a curriculum on the same day in either a directly preceding or successive time slot (i.e., in consecutive time slots). Each isolated lecture is counted as one violation.

Example 6.1. *Consider a curriculum containing courses C001 and C002, a day with six time slots, and the following schedules on that day:*

C001:		*R001*				
C002:				*R002*		

The schedules shows that course C001 has a lecture in the second time slot while C002 has a lecture in the fourth time slot. Since these time slots are not adjacent, this means that the schedules count two violations on that day. Consider now the following schedules:

C001:				*R001*		
C002:			*R002*			

Course C001 has a lecture in the fourth time slot while C002 has a lecture in the third one. These time slots are adjacent and these schedules do not count any violations on that day.

Minimum Working Days (MWD): For each course, a day is defined as a *working day* if at least one lecture is scheduled on it. For each course, a minimum number of working days is requested and each day below this minimum counts as one violation.

Room Capacity (RC): There is a number of students attending a given course and each room has a specific capacity. If the number of students assigned is greater than the room capacity, then every student that cannot be accommodated counts as one violation.

Room Stability (RStab): For every course, it is desired to schedule all lectures in just one room. Every additional room is counted as one violation.

As mentioned before, we only consider the scheduling of the courses into periods and ignore the assignment of rooms. Therefore we can only take the soft constraints **IL** and **MWD** into account which means that our approach is a relaxation of the original problem. Hence any valid lower bounds for our model are also valid lower bounds for the original problem.

It should be noted that the constraint **IL** is usually referred to as *curriculum compactness* in the literature. We have chosen to name it *isolated lectures* as multiple ways of formulating curriculum compactness are described in Bonutti et al. (2012) and this is the name they give for this formulation.

6.1.2 Related Work

As our focus is on lower bounds, we focus on the literature that have reported lower bounds as well. For the interested reader, we refer to Bettinelli et al. (2015) as they give more detailed descriptions of all the methods presented here as well as descriptions of many more methods including those to obtain upper bounds.

Burke et al. (2008), Burke et al. (2010), and Burke et al. (2012) describes a compact Integer Linear Programming (ILP) formulation which can be solved by a commercial ILP solver. They discover that this formulation is very hard to solve and tackle this in various ways. In Burke et al. (2010), one way to get a lower bound is to consider a relaxed version of the problem by setting the weights of the two room-related constraints **RC** and **RStab** to zero. This allows them to only consider the time-scheduling part of the problem. Furthermore a constraint is added to ensure that no more courses are scheduled in any time period than the total number of rooms available. Note that this is the same relaxation of the original problem that we are considering. They note that the before mentioned approach can be thought of as aggregating all the rooms into a single *multi-room*, where the capacity of this room is equal to the largest room. Another approach is to divide the rooms into sets of multi-rooms. For each multi-room, the capacity is equal to the largest room in the set and the maximum number of lectures that can take place in a multi-room in any period is equal to the number of rooms in the set. This allows them to partially include the room related soft constraints. After finding a solution to one of the before mentioned relaxations, the rooms are included in the model again and the solution obtained from the relaxation is used to guide the search by either fixing the periods that the lectures are assigned to or by fixing the days. In Burke et al. (2008), they enumerate all patterns for each day and for each curriculum and add the costs of these patterns as cuts. These cuts are also used in Burke et al. (2012) where they develop a cutting plane procedure and the cuts from Burke et al. (2008) are added dynamically as they are violated. Furthermore other cuts are derived including *clique cuts* and some more problem specific cuts. The reported results show that the cutting plane approach leads to better performances compared to the compact ILP formulation.

In Lach and Lübbecke (2012), the problem is decomposed into two stages: the first finds a feasible time-schedule, the second assigns the rooms to the lectures. This is based on their work in Lach and Lübbecke (2008) where they considered only hard constraints and showed that specific cuts could be added to the first stage problem to ensure feasible solutions. In Lach and Lübbecke (2012), they show how the **RC** constraint can be included in the first stage problem leaving only the **RStab** constraint for the second stage.

In Hao and Benlic (2011), the focus is on obtaining lower bounds for the first stage problem from Lach and Lübbecke (2012). Their approach is to decompose the courses and compute lower bounds for each of the decomposed sets of courses. A lower bound for the original problem is then calculated as the sum of the lower bounds of the decomposed sets of courses. The selection of the decomposition is embedded in a Tabu Search procedure (see Glover and Laguna, 2013).

Asín Achá and Nieuwenhuis (2012) encodes the constraints as satisfiability clauses. They apply different encodings both treating all constraints as hard and relaxing some or all of the soft constraints. They are able to prove optimality of more than half of the ITC2007 data sets.

Finally, Cacchiani et al. (2013) consider a model based on column generation. The idea is to split up the problem into two subproblems. One subproblem focuses on the room related soft constraints **RC** and **RO**. The other subproblem focuses on the time related soft constraints

MWD and **IL**. The overall lower bound is then the sum of the lower bounds obtained on the subproblems. They are able to obtain improved lower bounds and for three of the instances, they prove that the best known solutions are optimal.

Our approach is similar to Burke et al. (2008) in the sense that we enumerate all patterns for each day. The difference is that we apply them for the courses instead of the curricula and include them as variables instead of cuts. This gives us the possibility to make some preprocessing that may not be possible on the compact formulation and to derive valid inequalities from a conflict graph build on the pattern variables.

6.2 Pattern-based Formulation

In this section we present the pattern-based formulation. First the notation used throughout the article is given and then the mathematical model is described.

6.2.1 Notation

Throughout this article we refer to the following sets: $\mathcal{C}$ (courses), $\mathcal{L}$ (lecturers), $\mathcal{Q}$ (curricula), $\mathcal{D}$ (days), and $\mathcal{T}$ (time slots). Furthermore let $\mathcal{P}$ denote the set of periods, i.e., $\mathcal{P} = \mathcal{D} \times \mathcal{T}$. Then for each day $d \in \mathcal{D}$, the set $\mathcal{P}_d$ denotes the periods that belong to day d. The set of courses in curriculum $q \in \mathcal{Q}$ is $\mathcal{C}_q$, and similarly for a course $c \in \mathcal{C}$, the set of curricula that it belongs to is denoted $\mathcal{Q}_c$. The set of courses taught by lecturer $l \in \mathcal{L}$ is $\mathcal{C}_l$ and $l(c) \in \mathcal{L}$ refers to the lecturer teaching course c.

We also define a set Γ of course cliques. Consider a so-called *course conflict graph* that contains a node for every course. Two nodes are connected by an edge if they cannot be scheduled in the same period, i.e., if they are taught by the same lecturer or are in the same curriculum. As an example, consider the four courses c_1, c_2, c_3 and c_4. Let there be a curriculum consisting of the courses c_1, c_2, and c_3 and let there be a lecturer teaching courses c_3 and c_4. The corresponding course conflict graph is illustrated in Figure 6.1. All maximal cliques in Γ are enumerated (see Bron and Kerbosch, 1973), i.e., a clique $\gamma \in \Gamma$ is a set of courses $\mathcal{C}_\gamma$ where for each period at most one of the courses can have a lecture scheduled. Let $\mathcal{N}_c \subseteq \mathcal{C} \setminus \{c\}$ be the set of courses that is conflicting with $c \in \mathcal{C}$. In the conflict graph this corresponds to the neighborhood of the node representing c, i.e. all the nodes that are adjacent to c. As an example consider the node representing c_1 in Figure 6.1. The neighborhood $\mathcal{N}_{c_1}$ then consists of the nodes representing c_2 and c_3. Note that for any course $c_1 \in \mathcal{C}$ and $c_2 \in \mathcal{N}_{c_1}$ it must hold that $c_1 \in \mathcal{N}_{c_2}$.

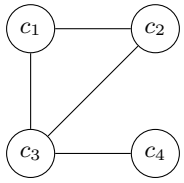


Figure 6.1: Conflict graph where a curriculum consists of courses c_1, c_2 and c_3 while c_3 and c_4 are taught by the same lecturer.

For a course $c \in \mathcal{C}$, the minimum number of days requested is denoted by the parameter $D_c^{\min}$. If course c is available in a specific period $p = (d, t) \in \mathcal{P}$, then we set the parameter $F_{c,d,t} = F_{c,p} = 1$ otherwise zero, and $\mathcal{P}_c$ refers to the set of periods where $F_{c,p} = 1$, i.e.,

$$\mathcal{P}_c = \{p \in \mathcal{P} \mid F_{c,p} = 1\}.$$

The number of lectures that needs to be scheduled for course c is denoted L_c and the total number of lectures to schedule for a clique $\gamma \in \Gamma$ is denoted L_γ, i.e.,

$$L_\gamma = \sum_{c \in \mathcal{C}_\gamma} L_c.$$

Lastly let R denote the number of available rooms and $W^{\mathbf{IL}}$ and $W^{\mathbf{MWD}}$ be the non-negative weights of violating the constraints **IL** and **MWD**, respectively. For $x \in \mathbb{R}$, let the function $(x)^+$ be defined by $(x)^+ := \max\{0, x\}$.

6.2.2 Mathematical Model

The pattern-based formulation aims at assigning a pattern to each day for each course. Given a pattern, the number of lectures scheduled in a pattern is independent on which day it is scheduled to. Therefore we generate all possible patterns based on the time slots $\mathcal{T}$. Consider the case $|\mathcal{T}| = 4$: sixteen different patterns exist which are all illustrated in Table 6.1. The first column refers to the time slots $t \in \mathcal{T}$ and the remaining columns refer to the patterns $k \in \mathcal{K}$. The symbol "×" in a cell means that the pattern contains the corresponding time slot. The last row (L_k) counts how many lectures pattern k contains.

Table 6.1: The patterns when $|\mathcal{T}| = 4$.

time slot	pattern index k															
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×				×	×	×				×	×	×		×
2			×			×			×	×		×	×		×	×
3				×			×		×		×	×		×	×	×
4					×			×		×	×		×	×	×	×
L_k	0	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4

For a pattern $k \in \mathcal{K}$ and a time slot $t \in \mathcal{T}$, the parameter a_t^k is set to one if pattern k contains a lecture in time slot t and zero otherwise. The set $\mathcal{P}_d^k \subseteq \mathcal{P}$ is the set of periods that some course is scheduled in if it is assigned to pattern k at day d. Based on this set, we need to define the feasible patterns $\mathcal{K}_{c,d}$ for each course c and for each day d:

$$\mathcal{K}_{c,d} = \left\{k \in \mathcal{K} \mid a_t^k \leq F_{c,d,t} \ \forall t \in \mathcal{T}\right\}, \quad c \in \mathcal{C}, d \in \mathcal{D}.$$

For each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, and pattern $k \in \mathcal{K}_{c,d}$, let $\lambda_{c,d}^k$ be a binary variable taking value one if course c is assigned to pattern k at day d, zero otherwise. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$, and time slot $t \in \mathcal{T}$, let binary variable $s_{q,d,t}$ take value one if curriculum q has an isolated lecture at day d in time slot t, zero otherwise. Finally, for each course $c \in \mathcal{C}$,

integer variable w_c counts the number of days below the requested minimum that course c is scheduled.

The pattern-based formulation of the curriculum-based course timetabling problem is as follows.

$$\min W^{\mathbf{IL}} \sum_{q\in\mathcal{Q},d\in\mathcal{D},t\in\mathcal{T}} s_{q,d,t} + W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c \tag{6.1}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}_{c,d}} \lambda^k_{c,d} = 1, \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{6.2}$$

$$\sum_{d\in\mathcal{D},k\in\mathcal{K}_{c,d}} L_k\, \lambda^k_{c,d} = L_c, \quad \forall c \in \mathcal{C} \tag{6.3}$$

$$\sum_{c\in\mathcal{C}_\gamma,k\in\mathcal{K}_{c,d}} a^k_t\, \lambda^k_{c,d} \le 1, \quad \forall \gamma \in \Gamma, d \in \mathcal{D}, t \in \mathcal{T} \tag{6.4}$$

$$\sum_{c\in\mathcal{C},k\in\mathcal{K}_{c,d}} a^k_t\, \lambda^k_{c,d} \le R, \quad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{6.5}$$

$$\sum_{d\in\mathcal{D},k\in\mathcal{K}_{c,d}:L_k\ge 1} \lambda^k_{c,d} + w_c \ge D^{\min}_c, \quad \forall c \in \mathcal{C} \tag{6.6}$$

$$\sum_{c\in\mathcal{C}_q,k\in\mathcal{K}_{c,d}} \left(a^k_t - \max\left\{a^k_{t-1}, a^k_{t+1}\right\}\right) \lambda^k_{c,d} \le s_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{6.7}$$

$$\lambda^k_{c,d} \in \mathbb{B}, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, k \in \mathcal{K}_{c,d} \tag{6.8}$$

$$s_{q,d,t} \in \mathbb{B}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{6.9}$$

$$w_c \in \mathbb{N}_0, \quad \forall c \in \mathcal{C} \tag{6.10}$$

The objective function (6.1) seeks to minimize a weighted sum of the violations of the soft constraints **IL** and **MWD**. Constraints (6.2) ensure that exactly one pattern is chosen for each course and day, whereas constraints (6.3) enforce the scheduling of all lectures of all courses. For every clique $\gamma \in \Gamma$, at most one lecture can be scheduled in any period as stipulated by constraints (6.4). The total number of rooms available is imposed by constraints (6.5). The number of violations for the soft constraints **IL** and **MWD** are computed through constraints (6.6) and (6.7), respectively. Finally, binary and integrality requirements (6.8)–(6.10) restrict the domains of the decision variables.

6.3 Preprocessing

In this section we show how some variables can be removed from the model and specifically how some patterns can never be feasible for some days and courses.

6.3.1 Simple Reductions

Some simple reductions can be made to model (6.1)–(6.10). Since a course $c \in \mathcal{C}$ must have L_c lectures scheduled, these require at least $\lceil L_c/|\mathcal{T}| \rceil$ days. So if the minimum working days

requested for a course c is less than or equal to this number, i.e., $D_c^{\min} \leq \lceil L_c/|\mathcal{T}| \rceil$, the constraint **MWD** can never be violated by that course and we can remove the variable w_c and the associated constraint (6.6). Another reduction is to consider a course c where $L_c = D_c^{\min}$, i.e., where the requested number of minimum working days is equal to the number of lectures. The only way for the course to not violate the constraint **MWD** is to only use the patterns containing at most one lecture. Assume that the course has chosen one lecture for each day. If the course then chooses to increase the number of lectures for one of the days, then it has to remove a lecture from one of the other days thus increasing the violation of the constraint **MWD** by one. This means that we can substitute the variable w_c throughout the model by the sum of the patterns containing at least two lectures where the coefficients are the number of lectures in the patterns minus one:

$$w_c = \sum_{\substack{d \in \mathcal{D}, \\ k \in \mathcal{K}_{c,d}: \\ L_k \geq 2}} (L_k - 1)\ \lambda_{c,d}^k, \quad \forall c \in \mathcal{C} : L_c = D_c^{\min}. \tag{6.11}$$

Another special case is when course c requests two working days, i.e., that $D_c^{\min} = 2$. The only way that the constraint **MWD** can be violated is when all lectures of the course are scheduled in a single day. Therefore the variable w_c can be substituted throughout the model by the sum of the patterns scheduling all lectures on one day:

$$w_c = \sum_{d \in \mathcal{D}, k \in \mathcal{K}_{c,d}: L_k = L_c} \lambda_{c,d}^k, \quad \forall c \in \mathcal{C} : D_c^{\min} = 2. \tag{6.12}$$

Some of the variables used for calculating the isolated lectures can also be substituted. Consider some curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$, and let $\mathcal{C}_{q,d,t}$ be the set of courses in the curriculum that can be scheduled in either that period or an adjacent period:

$$\mathcal{C}_{q,d,t} = \{c \in \mathcal{C}_q \mid F_{c,d,t-1} = 1 \vee F_{c,d,t} = 1 \vee F_{c,d,t+1} = 1\}. \tag{6.13}$$

If $|\mathcal{C}_{q,d,t}| = 1$ and the associated course chooses a pattern where a lecture is scheduled at day d in time slot t but not in an adjacent period, then this lecture must be an isolated lecture for q. This means we can substitute the variable $s_{q,d,t}$ in the objective function (6.1):

$$s_{q,d,t} = \sum_{\substack{c \in \mathcal{C}_q, k \in \mathcal{K}_{c,d}: \\ a_t^k = 1 \wedge a_{t-1}^k = a_{t+1}^k = 0}} \lambda_{c,d}^k, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} : |\mathcal{C}_{q,d,t}| = 1. \tag{6.14}$$

6.3.2 Pattern Elimination

When all patterns are generated, not all of them need to be considered for some course $c \in \mathcal{C}$. The total number of lectures to schedule for c is L_c. Clearly if some pattern contains more than L_c lectures, then this pattern can be excluded from all days. Furthermore, suppose that a pattern $k \in \mathcal{K}_{c,d}$ for some day $d \in \mathcal{D}$ has been selected. This means that no more patterns can be selected for day d so the remaining patterns for the days $\mathcal{D} \backslash \{d\}$ must be able to cover the remaining lectures $L_c - L_k$:

$$\sum_{d' \in \mathcal{D} \backslash \{d\}} \max_{k' \in \mathcal{K}_{c,d'}} \{L_{k'}\} \geq L_c - L_k, \quad \forall d \in \mathcal{D}, c \in \mathcal{C}, k \in \mathcal{K}_{c,d}. \tag{6.15}$$

If for any day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$ the condition (6.15) is not met, then this pattern is removed from that day. Similarly if for all days except d, patterns are chosen to cover a minimum number of lectures, then any pattern in $\mathcal{K}_{c,d}$ cannot cover more than the remaining lectures:

$$\sum_{d' \in \mathcal{D} \setminus \{d\}} \min_{k' \in \mathcal{K}_{c,d'}} \{L_{k'}\} \leq L_c - L_k, \quad \forall d \in \mathcal{D}, c \in \mathcal{C}, k \in \mathcal{K}_{c,d}. \tag{6.16}$$

So we scan through all patterns and remove the ones that do not fulfill conditions (6.15) and (6.16). If any patterns are removed during this scan, we iterate until no more patterns are removed.

The next elimination considers a course and checks if choosing a pattern would overlap too many feasible periods for some other courses. Consider the course $c_1 \in \mathcal{C}$ and the conflicting course $c_2 \in \mathcal{N}_{c_1}$. Course c_1 can at most be scheduled in $|\mathcal{P}_{c_2}| - L_{c_2}$ of the periods that are feasible for c_2 since all lectures must be scheduled:

$$\left|\mathcal{P}_{c_2} \cap \mathcal{P}_d^k\right| \leq |\mathcal{P}_{c_2}| - L_{c_2}, \quad \forall d \in \mathcal{D}, c_1 \in \mathcal{C}, c_2 \in \mathcal{N}_{c_1}, k \in \mathcal{K}_{c_1,d}. \tag{6.17}$$

The left-hand side in (6.17) is the number of periods that are feasible for c_2 and occupied by c_1 if pattern k is chosen. The right-hand side is the total number of feasible periods for c_2 minus the number of lectures that must be scheduled. Any pattern not fulfilling condition (6.17) is then removed from the model. This can be extended to the course cliques. Consider clique $\gamma \in \Gamma$ and a course $c \in \mathcal{C}_\gamma$ and let $\mathcal{P}_{\gamma \setminus c}$ be the feasible periods for all the courses in $\mathcal{C}_\gamma$ except c: $\mathcal{P}_{\gamma \setminus c} = \bigcup_{c' \in \mathcal{C}_\gamma \setminus \{c\}} \mathcal{P}_{c'}$. Any pattern chosen by c may not occupy more than the number of periods in $\mathcal{P}_{\gamma \setminus c}$ minus the sum of all the lectures to schedule for the courses in the clique except c since all lectures must be scheduled:

$$\left|\mathcal{P}_{\gamma \setminus c} \cap \mathcal{P}_d^k\right| \leq \left|\mathcal{P}_{\gamma \setminus c}\right| - \sum_{c' \in \mathcal{C}_\gamma \setminus \{c\}} L_{c'}, \quad \forall d \in \mathcal{D}, \gamma \in \Gamma, c \in \mathcal{C}_\gamma, k \in \mathcal{K}_{c,d}. \tag{6.18}$$

The left-hand side in (6.18) is the number of periods in the union of all feasible periods for the courses in the clique except c which is occupied by c if pattern k is selected. The right-hand side is the size of the union of the feasible periods of all the courses in the clique except c minus the total number of lectures that must be scheduled for these courses. Any pattern not fulfilling (6.18) are then removed.

The last elimination procedure solves a series of maximum flow problems. The idea is to consider a specific pattern for some course and then see if the remaining lectures in some clique can be scheduled. Consider a clique $\gamma \in \Gamma$, a pattern $k \in \mathcal{K}_{c,d}$ for some course $c \in \mathcal{C}_\gamma$ and day $d \in \mathcal{D}$. Let $m = L_k$, $n = |\mathcal{P}|$, and assume without loss of generality that the periods are numbered such that the first m periods, p_1, p_2, …, p_m, are those contained in pattern k. Let the next $n - m$ periods p_{m+1}, p_{m+2}, …, p_n be the remaining ones, i.e., all the periods that are not contained in the pattern. Since the course must choose exactly one pattern for each day, we know that if pattern k is selected then the course cannot be scheduled in the remaining periods on day d. Furthermore since we are considering a clique, we know that only the course under consideration can be scheduled in the first m periods. Assume that the courses are numbered such that the first $|\mathcal{C}_\gamma|$ are the courses that are in clique γ: c_1, c_2, …, $c_{|\mathcal{C}_\gamma|}$. Create a graph with a source node $\mathfrak{s}$ and a sink node $\mathfrak{t}$ (see Figure 6.2). An additional node $\mathfrak{s}$' is created to act as a dummy source of the non-selected periods. An arc with capacity $L_\gamma - L_k$ is added from

the source $\mathfrak{s}$ to the dummy source $\mathfrak{s}$' to model that the first L_k lectures must be scheduled by the source and the remaining $L_\gamma - L_k$ lectures must be scheduled by the dummy source. For each period p_1, p_2, ..., p_n, create a node and add an ingoing arc from the source $\mathfrak{s}$ if the period is in the set $\{p_1, p_2, \ldots, p_m\}$ and add an ingoing arc from the dummy source $\mathfrak{s}$' if the period is in the set $\{p_{m+1}, p_{m+2}, \ldots, p_n\}$. The capacities of all these arcs are set to one. For each course $c' \in \left\{c_1, c_2, \ldots, c_{|\mathcal{Q}_\gamma|}\right\}$, add a node to the graph and add an outgoing arc to the sink $\mathfrak{t}$ with a capacity of $L_{c'}$. For each node $p \in \{p_1, p_2, \ldots, p_n\}$ and each node $c' \in \left\{c_1, c_2, \ldots, c_{|\mathcal{C}_\gamma|}\right\}$, add an arc from p to c' with capacity $a\,(c', p)$ which is set depending on c and k:

$$a\,(c', p) = \begin{cases} F_{c',p} & \text{if } p \in \{p_1, \ldots, p_m\} \\ & \quad \text{and } c' = c \\ F_{c',p} & \text{if } p \in \{p_{m+1}, \ldots, p_n\} \\ & \quad \text{and } (c' \neq c \vee p \notin P_d) \\ 0 & \text{otherwise.} \end{cases} \tag{6.19}$$

If p is one of the first m periods, the capacity is set to $F_{c',p}$ if $c' = c$; if p is one of the $n - m$ last periods, the capacity is set to $F_{c',p}$ if p does not belong to day d or if $c' \neq c$. A maximum flow problem is solved and if the total amount of flow is less than the total number of lectures that need to be scheduled for the clique, then pattern k can never be feasible for course c at day d, and can thus be removed from $\mathcal{K}_{c,d}$.

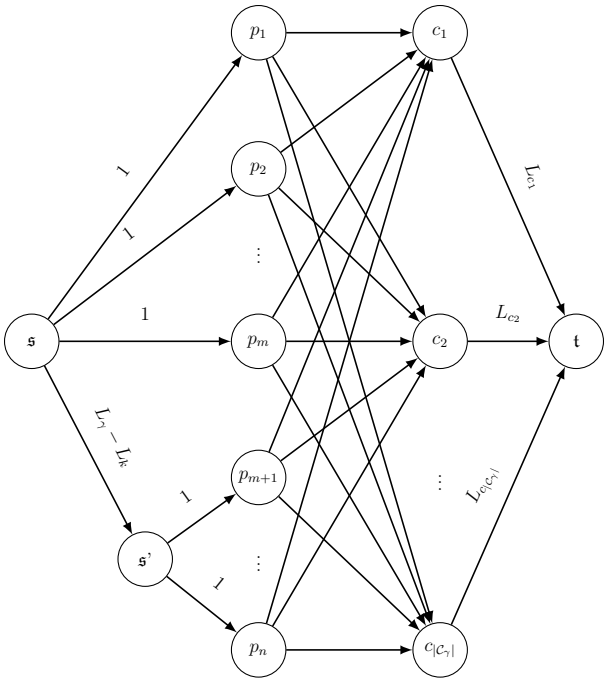


Figure 6.2: Given a clique γ, maximum flow graph to verify if pattern k for course $c \in \mathcal{C}_\gamma$ and day d can be discarded.

6.4 Valid Inequalities

In this section some strengthened bounds and valid inequalities are presented.

6.4.1 Implied Bounds

Burke et al. (2012) noted that since all lectures must be scheduled, then at least one day must be used by each course $c \in \mathcal{C}$ meaning that the constraint **MWD** can at most be violated by $D_c^{\min} - 1$. This can be strengthened a bit by noting that at most $|\mathcal{T}|$ lectures can be scheduled every day and so course c must be scheduled in at least $\lceil L_c/|\mathcal{T}| \rceil$ days:

$$0 \leq w_c \leq \left(D_c^{\min} - \left\lceil \frac{L_c}{|T|} \right\rceil \right)^+, \quad \forall c \in \mathcal{C}. \tag{6.20}$$

By considering the constraint **A**, this can be slightly improved. Let $\mathcal{D}_c = \left\{ d_1, d_2, \ldots, d_{|\mathcal{D}|} \right\}$ be the days for course c which are sorted such that $\sum_{t \in \mathcal{T}} F_{c,d_i,t} \geq \sum_{t \in \mathcal{T}} F_{c,d_j,t}$ for every $i, j \in \{1, 2, \ldots, |\mathcal{D}|\}$ where $i < j$. Then the minimum number of days that course c will be scheduled is the smallest index $i \in \{1, 2, \ldots, |\mathcal{D}|\}$ such that the total amount of feasible periods in the days $d_1, d_2, \ldots, d_i$ is enough to schedule all lectures of the course. We can then calculate an upper bound on the violation of the constraint **MWD**:

$$w_c^u = D_c^{\min} - \min_{i \in \{1,\ldots,|\mathcal{D}|\}} \left\{ i \;\middle|\; \sum_{\substack{j \in \{1,\ldots,i\}, \\ p \in \mathcal{P}_{d_j}}} F_{c,p} \geq L_c \right\}, \quad \forall c \in \mathcal{C}. \tag{6.21}$$

It is also possible to calculate the maximum number of days course c can be scheduled in. Obviously it is not possible to schedule more than L_c days. Using the constraint **A**, it is not possible to schedule more days than the set of days where at least one of the time slots is feasible for the course. So a lower bound on the violation of the constraint **MWD** can also be calculated:

$$w_c^l = D_c^{\min} - \max \left\{ L_c, \left| \left\{ d \in \mathcal{D} \;\middle|\; \sum_{p \in \mathcal{P}_d} F_{c,p} \geq 1 \right\} \right| \right\}, \quad \forall c \in \mathcal{C}. \tag{6.22}$$

This gives the following bounds:

$$\left(w_c^l \right)^+ \leq w_c \leq \left(w_c^u \right)^+, \quad \forall c \in \mathcal{C}. \tag{6.23}$$

Furthermore since we have just found the minimum and maximum number of days that a course can have scheduled lectures, we can add the following inequalities to model (6.1)–(6.10):

$$\sum_{\substack{d \in \mathcal{D}, \\ k \in \mathcal{K}_{c,d}: \\ L_k \geq 1}} \lambda_{c,d}^k \geq \min_{i \in \{1,\ldots,|\mathcal{D}|\}} \left\{ i \;\middle|\; \sum_{\substack{j \in \{1,\ldots,i\}, \\ p \in \mathcal{P}_{d_j}}} F_{c,p} \geq L_c \right\}, \quad \forall c \in \mathcal{C}. \tag{6.24}$$

$$\sum_{\substack{d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}:\\ L_k\geq 1}} \lambda_{c,d}^k \leq \max\left\{L_c, \left|\left\{d\in\mathcal{D}\,|\,\sum_{p\in\mathcal{P}_d} F_{c,p}\geq 1\right\}\right|\right\}, \quad \forall c\in\mathcal{C}. \tag{6.25}$$

6.4.2 Extended Cover Inequalities

Consider clique $\gamma\in\Gamma$. Since we must schedule the lectures for all courses, trivially the following must hold:

$$\sum_{c\in\mathcal{C}_\gamma, d\in\mathcal{D}, k\in\mathcal{K}_{c,d}} L_k\,\lambda_{c,d}^k \leq L_\gamma, \quad \forall\gamma\in\Gamma. \tag{6.26}$$

Note that the conditions (6.26) are knapsack constraints. This means that we can apply known valid inequalities such as *cover inequalities*, see e.g. Van Roy and Wolsey (1987). As an example, consider some clique $\gamma\in\Gamma$ with $L_\gamma = 5$. Assume we only choose patterns containing two lectures, then surely we cannot choose more than two of these patterns otherwise the knapsack constraint is violated. This means that if we choose three patterns, e.g. $\lambda_{c_1,d_1}^{k_1}$, $\lambda_{c_2,d_2}^{k_2}$ and $\lambda_{c_3,d_3}^{k_3}$ where $L_{k_1} = L_{k_2} = L_{k_3} = 2$, then we have a cover and we can add the *cover inequality*:

$$\lambda_{c_1,d_1}^{k_1} + \lambda_{c_2,d_2}^{k_2} + \lambda_{c_3,d_3}^{k_3} \leq 2.$$

We can then add each variable to the inequality where $L_k \geq 2$ creating an *extended cover inequality*. This can be generalized to the following inequalities:

$$\sum_{\substack{c\in\mathcal{C}_\gamma, d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}: L_k\geq i}} \lambda_{c,d}^k \leq \left\lfloor\frac{L_\gamma}{i}\right\rfloor, \quad \forall\gamma\in\Gamma, i\in\left\{2,\ldots,\left\lfloor\frac{L_\gamma}{2}\right\rfloor+1\right\}. \tag{6.27}$$

These inequalities consider that each clique $\gamma\in\Gamma$ can at most be scheduled for L_γ lectures. Since all lectures must be scheduled, L_γ is also a lower bound on the number of lectures that must be scheduled:

$$\sum_{c\in\mathcal{C}_\gamma, d\in\mathcal{D}, k\in\mathcal{K}_{c,d}} L_k\,\lambda_{c,d}^k \geq L_\gamma, \quad \forall\gamma\in\Gamma. \tag{6.28}$$

To be able to add the *extended cover inequalities* by using constraints (6.28), we need to reformulate it into a knapsack constraint similar to (6.26). To do this, we first define $L_{c,d}^{\max}$ as the maximum number of lectures that course $c\in\mathcal{C}$ can be scheduled on day $d\in\mathcal{D}$:

$$L_{c,d}^{\max} := \max_{k\in\mathcal{K}_{c,d}}\{L_k\}. \tag{6.29}$$

Taking constraints (6.2) and multiplying by $L_{c,d}^{\max}$ gives us the following:

$$\sum_{k\in\mathcal{K}_{c,d}} L_{c,d}^{\max}\,\lambda_{c,d}^k = L_{c,d}^{\max}, \quad \forall c\in\mathcal{C}, d\in\mathcal{D}. \tag{6.30}$$

Obviously we can add $L_k - L_k$ to the coefficient of $\lambda_{c,d}^k$ in (6.30) and rearrange the terms:

$$\sum_{k \in \mathcal{K}_{c,d}} \left(L_k + L_{c,d}^{\max} - L_k\right) \lambda_{c,d}^k = L_{c,d}^{\max}, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}. \tag{6.31}$$

$$\sum_{k \in \mathcal{K}_{c,d}} L_k \lambda_{c,d}^k = L_{c,d}^{\max} - \sum_{k \in \mathcal{K}_{c,d}} \left(L_{c,d}^{\max} - L_k\right) \lambda_{c,d}^k, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}. \tag{6.32}$$

Substituting into (6.28) and rearranging the terms gives the following:

$$\sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}} \left(L_{c,d}^{\max} - \sum_{k \in \mathcal{K}_{c,d}} \left(L_{c,d}^{\max} - L_k\right) \lambda_{c,d}^k \right) \geq L_\gamma, \quad \forall \gamma \in \Gamma. \tag{6.33}$$

$$\sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}, k \in \mathcal{K}_{c,d}} \left(L_{c,d}^{\max} - L_k\right) \lambda_{c,d}^k \leq \sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}} L_{c,d}^{\max} - L_\gamma, \quad \forall \gamma \in \Gamma. \tag{6.34}$$

Since $L_{c,d}^{\max}$ is the maximum number of lectures that can be scheduled for course c on day d, then $L_{c,d}^{\max} \geq L_k$ for every pattern $k \in \mathcal{K}_{c,d}$. Furthermore we assume that $\sum_{d \in \mathcal{D}} L_{c,d}^{\max} \geq L_c$ for each course c, otherwise it can easily be seen that the problem would be infeasible. Hence constraints (6.34) are the wanted knapsack constraints. For ease of notation, we define:

$$\overline{L}_c^k := L_{c,d}^{\max} - L_k, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, k \in \mathcal{K}_{c,d}. \tag{6.35}$$

$$\overline{L}_\gamma := \sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}} L_{c,d}^{\max} - L_\gamma, \quad \forall \gamma \in \Gamma. \tag{6.36}$$

We can then add the *extended cover inequalities* in the same way as the inequalities (6.27) for constraints (6.26):

$$\sum_{\substack{c \in \mathcal{C}_\gamma, d \in \mathcal{D}, \\ k \in \mathcal{K}_{c,d} : \overline{L}_c^k \geq i}} \lambda_{c,d}^k \leq \left\lfloor \frac{\overline{L}_\gamma}{i} \right\rfloor, \quad \forall \gamma \in \Gamma, i \in \left\{2, \ldots, \left\lfloor \frac{\overline{L}_\gamma}{2} \right\rfloor + 1\right\}. \tag{6.37}$$

Not all of *extended cover inequalities* are necessarily added. Consider some clique $\gamma \in \Gamma$ and
$i, j \in \{2, \ldots, \lfloor L_\gamma/2 \rfloor + 1\}$ where $i < j$. If $\lfloor L_\gamma/i \rfloor = \lfloor L_\gamma/j \rfloor$, constraint (6.27) with respect to j is redundant and is not added to the model. The same argument can be used for constraints (6.37).

6.4.3 The Pattern Conflict Graph and Inequalities

The inequalities we focus on in this section are based on creating a *pattern conflict graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$, we create a node $v_{c,d}^k \in \mathcal{V}$. Two nodes $v_{c1,d1}^{k1}, v_{c2,d2}^{k2} \in \mathcal{V}$ are connected with an edge if the corresponding patterns cannot both be included in a feasible solution, i.e., if they are *conflicting*. We then use this graph to derive valid inequalities. Some of these inequalities are clique inequalities derived by finding the cliques in the pattern conflict graph. This graph can become very large so it is not practical to enumerate all the maximal cliques as we did in Section 6.2 for the *course conflict graph*. Instead we run a heuristic aimed at minimizing the number of cliques needed to

cover all edges as described by Kou et al. (1978). Let Θ be the set of cliques returned by the heuristic and let $\mathcal{V}_\theta \subseteq \mathcal{V}$ be the set of nodes in clique $\theta \in \Theta$. We then add the following clique inequalities:

$$\sum_{v^k_{c,d} \in \mathcal{V}_\theta} \lambda^k_{c,d} \leq 1, \quad \forall \theta \in \Theta. \tag{6.38}$$

The next set of valid inequalities is based on constraints (6.7) and inspired by the *Lifted Class Compactness* inequalities described in Avella and Vasil'ev (2005, Proposition 5.4). In the left-hand side of constraint (6.7) for curriculum $q \in \mathcal{Q}$ and time slot $t \in \mathcal{T}$, let $\overline{a}^k_t$ be the coefficient of the pattern variable $\lambda^k_{c,d}$ for course $c \in \mathcal{C}_q$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$:

$$\overline{a}^k_t := a^k_t - \max\{a^k_{t-1}, a^k_{t+1}\} = \begin{cases} 1 & \text{if } a^k_t = 1 \wedge a^k_{t-1} = a^k_{t+1} = 0 \\ -1 & \text{if } a^k_t = 0 \wedge \left(a^k_{t-1} = 1 \vee a^k_{t+1} = 1\right) \\ 0 & \text{otherwise.} \end{cases} \tag{6.39}$$

Let $\mathcal{V}^+_{q,d,t}$ and $\mathcal{V}^-_{q,d,t}$ be the sets of nodes representing the patterns for which the coefficient is equal to one and to minus one, respectively:

$$\mathcal{V}^+_{q,d,t} := \left\{v^k_{c,d} \in \mathcal{V} \,|\, c \in \mathcal{C}_q, k \in \mathcal{K}_{c,d}, \overline{a}^k_t = 1\right\}. \tag{6.40}$$

$$\mathcal{V}^-_{q,d,t} := \left\{v^k_{c,d} \in \mathcal{V} \,|\, c \in \mathcal{C}_q, k \in \mathcal{K}_{c,d}, \overline{a}^k_t = -1\right\}. \tag{6.41}$$

Based on these sets, we can reformulate constraints (6.7):

$$\sum_{v^k_{c,d} \in \mathcal{V}^+_{q,d,t}} \lambda^k_{c,d} - \sum_{v^k_{c,d} \in \mathcal{V}^-_{q,d,t}} \lambda^k_{c,d} \leq s_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}. \tag{6.42}$$

In constraints (6.42), it can be seen that for a curriculum $q \in \mathcal{Q}$, there is an isolated lecture at day d and time slot t if one of the patterns in $\mathcal{V}^+_{q,d,t}$ is selected and none of the patterns in $\mathcal{V}^-_{q,d,t}$ are selected. Consider now some clique $\mathcal{H}_{q,d,t} \subseteq \mathcal{V}$, where every node in $\mathcal{H}_{q,d,t}$ is conflicting with every node in $\mathcal{V}^-_{q,d,t}$. If any pattern in $\mathcal{H}_{q,d,t}$ is included in a solution, then none of the patterns in $\mathcal{V}^-_{q,d,t}$ can be chosen. This means that if a pattern has been chosen from the set $\mathcal{V}^+_{q,d,t}$ and a pattern has been chosen from the set $\mathcal{H}_{q,d,t}$, then curriculum q must have an isolated lecture at day d and time slot t. Therefore the following *adjacent isolated lecture inequalities* can be added:

$$\sum_{v^k_{c,d} \in \mathcal{V}^+_{q,d,t}} \lambda^k_{c,d} + \sum_{v^k_{c,d'} \in \mathcal{H}_{q,d,t}} \lambda^k_{c,d} - 1 \leq s_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}. \tag{6.43}$$

The way we generate a clique $\mathcal{H}_{q,d,t}$ is to take the subgraph induced by the neighbors of $\mathcal{V}^-_{q,d,t}$. We then run the heuristic described by Kou et al. (1978) to get a clique cover of the neighbors and, for each clique returned by the heuristic, an inequality (6.43) is added, where the clique defines set $\mathcal{H}_{q,d,t}$.

6.4.4 Generating Edges for the Pattern Conflict Graph

The pattern conflict graph was briefly described in Section 6.4.3. Here we explain how we derive the edges, i.e., how we figure out which of the patterns are conflicting. The first set of edges are derived from constraints (6.2). Since at most one pattern can be chosen for each course and each day, we can add an edge between nodes $v_{c,d}^{k_1}$ and $v_{c,d}^{k_2}$ for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, and patterns $k_1 \in \mathcal{K}_{c,d}$ and $k_2 \in \mathcal{K}_{c,d} \backslash \{k_1\}$. Next we can also add edges between nodes $v_{c,d_1}^{k_1}$ and $v_{c,d_1}^{k_2}$ if $L_{k_1} + L_{k_2} > L_c$.

Another set of edges comes from the constraint **C**. For two conflicting courses $c_1, c_2 \in \mathcal{C}$, we add an edge between nodes $v_{c_1,d}^{k_1}$ and $v_{c_2,d}^{k_2}$ for day $d \in \mathcal{D}$ and patterns $k_1 \in \mathcal{K}_{c_1,d}$ and $k_2 \in \mathcal{K}_{c_2,d}$ if there exists some time slot contained in both patterns, i.e., if $\exists t \in \mathcal{T} : a_t^{k_1} = a_t^{k_2} = 1$.

The next set of edges is derived by extending the methods to eliminate the patterns as discussed in Section 6.3, where instead of checking for infeasibility of selecting only one pattern, we check for infeasibility of selecting a pair of patterns. For example, it was checked in constraint (6.15) whether selecting a specific pattern would make it possible to cover the remaining lectures on the other days. Consider some course $c \in \mathcal{C}$, two days $d_1, d_2 \in \mathcal{D}$, and two patterns $k_1 \in \mathcal{K}_{c,d_1}$ and $k_2 \in \mathcal{K}_{c,d_2}$ where $d_1 \neq d_2$. If both patterns are selected, then it must be possible to cover the remaining $L_c - L_{k_1} - L_{k_2}$ lectures from the patterns associated with the days in set $\mathcal{D}\backslash\{d_1, d_1\}$. We add an edge between nodes $v_{c,d_1}^{k_1}$ and $v_{c,d_2}^{k_2}$ if the following condition is not met:

$$\sum_{d \in \mathcal{D}\backslash\{d_1,d_2\}} \max_{k \in \mathcal{K}_{c,d}} \{L_k\} \geq L_c - L_{k_1} - L_{k_2}. \tag{6.44}$$

Similarly if the two patterns $k_1 \in \mathcal{K}_{c,d_1}$ and $k_2 \in \mathcal{K}_{c,d_2}$ are chosen, then the patterns containing the smallest number of lectures on the other days cannot exceed the remaining lectures. We add an edge between nodes $v_{c,d_1}^{k_1}$ and $v_{c,d_2}^{k_2}$ if the following condition is not met:

$$\sum_{d \in \mathcal{D}\backslash\{d_1,d_2\}} \min_{k \in \mathcal{K}_{c,d}} \{L_k\} \leq L_c - L_{k_1} - L_{k_2}. \tag{6.45}$$

We can also verify if selecting a pair of patterns would overlap too many feasible periods for some courses as in (6.17) and (6.18). First all pairs of conflicting courses are considered. Let $c_1, c_2 \in \mathcal{C}$ be conflicting. If pattern $k_2 \in \mathcal{K}_{c_2,d_2}$ is chosen for course c_2 on day $d_2 \in \mathcal{D}$, we know that no other pattern can be chosen for course c_2 for that day. Therefore it is not possible to choose a pattern $k_1 \in \mathcal{K}_{c_1,d_1}$ for course c_1 on day $d_1 \in \mathcal{D}\backslash\{d_2\}$ if that pattern occupies more than $L_{c_2} - L_{k_2}$ of the feasible periods for c_2 in the days $\mathcal{D}\backslash\{d_2\}$. Hence, for the pair of nodes $v_{c_1,d_1}^{k_1}$ and $v_{c_2,d_2}^{k_2}$, we add an edge if the following condition does not hold:

$$\left|\mathcal{P}_{c_2} \cap \mathcal{P}_{d_1}^{k_1}\right| \leq \left|\mathcal{P}_{c_2} \setminus \mathcal{P}_{d_2}\right| - L_{c_2} + L_{k_2}. \tag{6.46}$$

Consider now two courses $c_1, c_2 \in \mathcal{C}$ that need not be conflicting nor distinct. In the course conflict graph, we look at the nodes representing c_1 and c_2 and the neighborhood of these two nodes, i.e., the nodes that are connected by an edge to both c_1 and c_2. Let $\mathcal{N}_{c_1,c_2}$ be this neighborhood (illustrated in Figure 6.3) and defined as

$$\mathcal{N}_{c_1,c_2} = \mathcal{N}_{c_1} \cap \mathcal{N}_{c_2}. \tag{6.47}$$

It is not possible to assign a pair of patterns to courses c_1 and c_2 that together occupy more

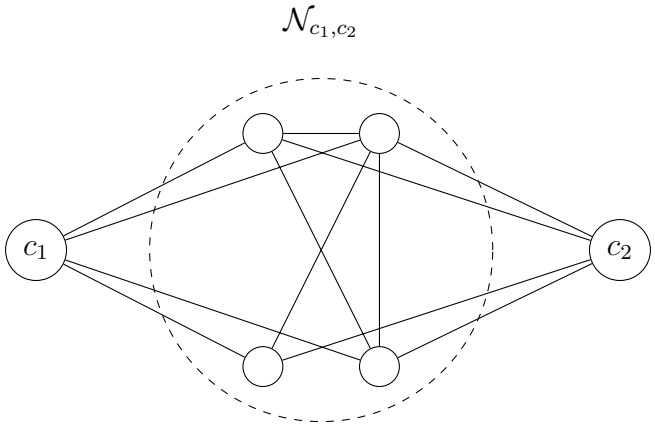


Figure 6.3: Neighborhood $\mathcal{N}_{c_1,c_2}$ of the courses $c_1, c_2 \in \mathcal{C}$ in the course clique graph.

than $|\mathcal{P}_{c_3}| - L_{c_3}$ of the feasible periods of $c_3 \in \mathcal{N}_{c_1,c_2}$. Therefore we get the following conditions:

$$\left|\mathcal{P}_{c_3} \cap \left(\mathcal{P}_{d_1}^{k_1} \cup \mathcal{P}_{d_2}^{k_2}\right)\right| \leq |\mathcal{P}_{c_3}| - L_{c_3}. \tag{6.48}$$

Hence for each pair of nodes $v_{c_1,d_1}^{k_1}$ and $v_{c_2,d_2}^{k_2}$, we add an edge if conditions (6.48) are not met. This can be extended by considering a pair of nodes in the neighborhood of c_1 and c_2 that are themselves conflicting. Let $c_3, c_4 \in \mathcal{N}_{c_1,c_2}$ be such a pair, i.e., $c_4 \in \mathcal{N}_{c_3}$. It is not feasible to assign to courses c_1 and c_2 patterns that together occupy more than $|\mathcal{P}_{c_3} \cup \mathcal{P}_{c_4}| - L_{c_3} - L_{c_4}$ of the feasible periods for c_3 and c_4 because all of their lectures must be scheduled in distinct periods. The following conditions must be met:

$$\left|(\mathcal{P}_{c_3} \cup \mathcal{P}_{c_4}) \cap \left(\mathcal{P}_{d_1}^{k_1} \cup \mathcal{P}_{d_2}^{k_2}\right)\right| \leq |\mathcal{P}_{c_3} \cup \mathcal{P}_{c_4}| - L_{c_3} - L_{c_4}. \tag{6.49}$$

We can extend it even further by considering cliques in the neighborhood of c_1 and c_2. Let Γ_{c_1,c_2} be a set of cliques of the neighborhood $\mathcal{N}_{c_1,c_2}$ found by iterating through each clique $\gamma \in \Gamma$ and then add the courses $\mathcal{C}' = \mathcal{N}_{c_1,c_2} \cap \mathcal{C}_\gamma$ if $\mathcal{C}' \neq \emptyset$:

$$\Gamma_{c_1,c_2} = \left\{\mathcal{C}' \subseteq \mathcal{N}_{c_1,c_2} \,|\, \exists \gamma \in \Gamma : \mathcal{C}' \subseteq \mathcal{C}_\gamma\right\}. \tag{6.50}$$

As for a clique in Γ, we also refer to the courses in $\gamma \in \Gamma_{c_1,c_2}$ by $\mathcal{C}_\gamma$. The set of feasible periods of all the courses in $\mathcal{C}_\gamma$ is denoted by $\mathcal{P}_\gamma^{c_1,c_2}$:

$$\mathcal{P}_\gamma^{c_1,c_2} = \bigcup_{c \in \mathcal{C}_\gamma} \mathcal{P}_c. \tag{6.51}$$

Note that $\mathcal{P}_\gamma^{c_1,c_2}$ does not necessarily include the feasible periods for c_1 and c_2 as they are not in the set of courses $\mathcal{C}_\gamma$. We add an edge between nodes $v_{c_1,d_2}^{k_1}$ and $v_{c_2,d_2}^{k_2}$ if the following conditions are not met:

$$\left|\mathcal{P}_\gamma^{c_1,c_2} \cap \left(\mathcal{P}_{d_1}^{k_1} \cup \mathcal{P}_{d_2}^{k_2}\right)\right| \leq \left|\mathcal{P}_\gamma^{c_1,c_2}\right| - \sum_{c' \in \mathcal{C}_\gamma} L_{c'}. \tag{6.52}$$

The last method to identify edges in the pattern conflict graph is to solve a series of maximum flow problems. The maximum flow graph is an extension to the graph used for eliminating

patterns in Section 6.3.2. Consider a clique $\gamma \in \Gamma$, a pattern $k_1 \in \mathcal{K}_{c_1,d_1}$ for some course $c_1 \in \mathcal{C}_\gamma$ and day $d_1 \in \mathcal{D}$ and a pattern $k_2 \in \mathcal{K}_{c_2,d_2}$ for some course $c_2 \in \mathcal{C}_\gamma$ and day $d_2 \in \mathcal{D}$. Let $m_1 = L_{k_1}$, $m_2 = L_{k_2} + L_{k_1}$ and $n = |\mathcal{P}|$ and assume that the periods are numbered such that the first m_1 periods p_1, p_2, ..., p_{m_1} are the periods contained in pattern k_1 and the next $m_2 - m_1$ periods p_{m_1+1}, p_{m_1+2}, ..., p_{m_2} are the periods contained in pattern k_2. Let the next $n - m_2$ periods p_{m_2+1}, p_{m_2+2}, ..., p_n be the remaining periods, i.e., all the periods that are not contained in the patterns. Since exactly one pattern must be chosen for each course c_1 and c_2 and each day, we know that if patterns k_1 and k_2 are selected, then the courses cannot be scheduled in the remaining periods on day d_1 and d_2, respectively. Furthermore since we are considering a clique, we know that only c_1 can be scheduled in the first m_1 periods and only c_2 can be scheduled in the next $m_2 - m_1$ periods. Assume that the courses are numbered such that the first $|\mathcal{C}_\gamma|$ are the courses in clique γ: c_1, c_2, ..., $c_{|\mathcal{C}_\gamma|}$. Create a graph with a source node $\mathfrak{s}$, a dummy source node $\mathfrak{s}$', and a sink node $\mathfrak{t}$ (see Figure 6.4). An arc with capacity $L_\gamma - L_{k_1} - L_{k_2}$ is added from the source $\mathfrak{s}$ to the dummy source $\mathfrak{s}$'. For each period p_1, p_2, ..., p_n, create a node and add an ingoing arc from the source $\mathfrak{s}$ if the period is in the set $\{p_1, p_2, \ldots, p_{m_2}\}$ and an ingoing arc from the dummy source $\mathfrak{s}$' if the period is in the set $\{p_{m_2+1}, p_{m_2+2}, \ldots, p_n\}$. The capacities of all these arcs are set to one. For each course $c' \in \{c_1, c_2, \ldots, c_{|\mathcal{Q}_\gamma|}\}$, add a node to the graph and an outgoing arc to the sink $\mathfrak{t}$ with a capacity of $L_{c'}$. For each node $c' \in \{c_1, c_2, \ldots, c_{|\mathcal{C}_\gamma|}\}$ and for each node $p \in \{p_1, p_2, \ldots, p_n\}$, add an arc from p to c' with capacity $a(p, c')$ defined as follows:

$$a(p,c') = \begin{cases} F_{c',p} & \text{if } p \in \{p_1, \ldots, p_{m_1}\} \\ & \quad \text{and } c' = c_1 \\ F_{c',p} & \text{if } p \in \{p_{m_1+1}, \ldots, p_{m_2}\} \\ & \quad \text{and } c' = c_2 \\ F_{c',p} & \text{if } p \in \{p_{m_2+1}, \ldots, p_n\} \\ & \quad \text{and } (c' \neq c_1 \vee p \notin P_{d_1}) \\ & \quad \text{and } (c' \neq c_2 \vee p \notin P_{d_2}) \\ 0 & \text{otherwise.} \end{cases} \tag{6.53}$$

If p is one of the first m_1 periods, the capacity is set to $F_{c',p}$ if $c' = c_1$; if p is one of the next $m_2 - m_1$ periods, the capacity is set to $F_{c',p}$ if $c' = c_2$; if p is one of the $n - m_2$ last periods, the capacity is set to $F_{c',p}$ if p does not belong to day d_1 or if $c' \neq c_1$ and if p does not belong to d_2 or $c' \neq c_2$. A maximum flow problem is then solved and if the total amount of flow is less than L_γ, then the two patterns cannot both be selected in a feasible solution and an edge in the conflict graph is added between $v^{k_1}_{c_1,d_2}$ and $v^{k_2}_{c_2,d_2}$.

Even if the maximum flow problem can be solved efficiently, we need to solve many problems, resulting in a long total running time. This can be speeded up by, instead of creating a graph for each pair of nodes in the conflict graph, we create the graph for an entire clique and then adjust the capacities according to the courses and patterns that are under consideration. Consider a clique $\gamma \in \Gamma$ and create the same nodes as above, i.e., the source node $\mathfrak{s}$, the dummy source node $\mathfrak{s}$', the sink $\mathfrak{t}$, a node for each period and a node for each course. As before, each node representing a course $c \in \mathcal{C}_\gamma$ has an outgoing arc connected to the sink $\mathfrak{t}$ with a capacity of L_c. There is an arc from the source $\mathfrak{s}$ to the dummy source $\mathfrak{s}$', but this time the capacity is initially set to zero. Each period node has an incoming arc from both the source $\mathfrak{s}$ and the

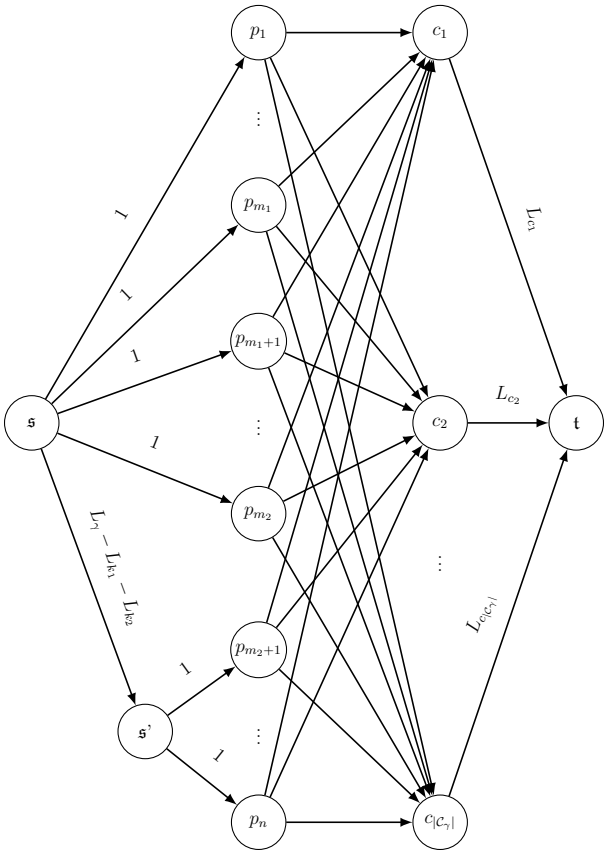


Figure 6.4: Given a clique $\gamma \in \Gamma$, maximum flow graph to identify if patterns $k_1 \in \mathcal{K}_{c_1,d_1}$ and $k_2 \in \mathcal{K}_{c_2,d_2}$ for some courses $c_1, c_2 \in \mathcal{C}_\gamma$ and days $d_1, d_2 \in \mathcal{D}$ cannot both be contained in a feasible solution.

dummy source $\mathfrak{s}$' and has an outgoing arc to each course $c \in \mathcal{C}_\gamma$ that can be feasibly scheduled in that period. The capacities of all the arcs entering and leaving the periods are initially set to zero. All capacities that are initially set to zero are adjusted according to the courses, days and patterns that are under consideration. At first the flow on all arcs is zero. This flow is a maximum integer flow since all arcs leaving the source have zero capacity, no capacities are violated and all node balancing constraints are satisfied. After this, the approach is then to run an iterative algorithm that maintains a maximum integer flow as the loop invariant. Each iteration consists of three steps that first picks two patterns to investigate and adjust the capacities accordingly, possibly creating an infeasible flow. The second step is to repair the possibly infeasible flow created by the adjustment of the capacities. The last step is to recalculate the maximum flow.

Step 1 – Select Patterns and Adjust Capacities. Select courses $c_1, c_2 \in \mathcal{C}_\gamma$, days $d_1, d_2 \in \mathcal{D}$ and patterns $k_1 \in \mathcal{K}_{c_1,d_1}$ and $k_2 \in \mathcal{K}_{c_2,d_2}$ which have not been considered yet and where there is no edge between $v^{k_1}_{c_1,d_1}$ and $v^{k_2}_{c_2,d_2}$ in the pattern conflict graph. The capacities are then adjusted according to the selected patterns. First the capacity on the arc from the source $\mathfrak{s}$ to the dummy $\mathfrak{s}$' is set to the total amount of lectures that need to be scheduled for the clique

minus the amount of lectures in the two patterns:

$$a\left(\mathfrak{s}, \mathfrak{s}'\right) = L_\gamma - L_{k_1} - L_{k_2}. \tag{6.54}$$

The capacities from the source $\mathfrak{s}$ to a node for some period p are set to one if the period is part of any of the two patterns, otherwise it is set to zero:

$$a\left(\mathfrak{s}, p = (d,t)\right) = \begin{cases} 1 & \text{if } d = d_1 \wedge a_t^{k_1} = 1 \\ 1 & \text{if } d = d_2 \wedge a_t^{k_2} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6.55}$$

The capacities from $\mathfrak{s}$' to a node for some period p are the opposite of the arcs from the source $\mathfrak{s}$:

$$a\left(\mathfrak{s}', p = (d,t)\right) = \begin{cases} 0 & \text{if } d = d_1 \wedge a_t^{k_1} = 1 \\ 0 & \text{if } d = d_2 \wedge a_t^{k_2} = 1 \\ 1 & \text{otherwise.} \end{cases} \tag{6.56}$$

Capacities (6.54), (6.55) and (6.56) together ensure that the maximum flow can only be equal to the total number of lectures if some flow is sent through all the periods from the selected patterns. One must also ensure that the flow leaving the periods from the selected patterns are only sent to the designated courses, i.e., for each period $p = (d,t)$, if $d = d_1 \wedge a_t^{k_1} = 1$, the flow can only be sent to c_1 and if $d = d_2 \wedge a_t^{k_2} = 1$, the flow can only be sent to c_2. Furthermore, since one pattern must be selected for each course and each day, then c_1 cannot be scheduled in any period at day d_1 which is not contained in pattern k_1; the same holds true for course c_2, day d_2 and pattern k_2. This results in the following capacities:

$$a\left(p = (d,t), c\right) = \begin{cases} F_{c,p}\, a_t^{k_1} & \text{if } d = d_1 \wedge c = c_1 \\ F_{c,p}\, a_t^{k_2} & \text{if } d = d_2 \wedge c = c_2 \\ F_{c,p}\left(1 - a_t^{k_1}\right) & \text{if } d = d_1 \wedge c \neq c_1 \\ F_{c,p}\left(1 - a_t^{k_2}\right) & \text{if } d = d_2 \wedge c \neq c_2 \\ F_{c,p} & \text{otherwise.} \end{cases} \tag{6.57}$$

Step 2 – Repair Flow. If the capacity adjustment from Step 1 is making the flow infeasible, then the flow is adjusted to regain feasibility. Let $f(u,v)$ be the flow on arc (u,v) and f_{viol} be the sum of all the excess flow on the arcs:

$$f_{\text{viol}} = \sum_{(u,v)} \left(f(u,v) - a(u,v)\right)^+.$$

The adjustments are done by going through all the arcs in the flow graph. If the capacity is violated on an arc (u,v), it must be because the amount of flow is at least one unit more than the capacity: $f(u,v) \geq a(u,v) + 1$. This is because we know from the loop invariant that the flow is integral and because the capacities are either changed from one to zero or from zero to one. This means that there must exist some directed path going from the source $\mathfrak{s}$ to the sink $\mathfrak{t}$ through arc (u,v) with a flow of at least one unit on all its arcs. Such a path is illustrated in Figure 6.5 and can easily be found by identifying two shortest paths (with respect to the

number of arcs), from $\mathfrak{s}$ to u and from v to $\mathfrak{t}$ using only arcs where there are at least one unit of flow. These paths are then put together into one path with the arc (u, v). Let $\Delta^{\mathfrak{s},\mathfrak{t}}_{(u,v)} \geq 1$ be the smallest excess flow on that path, we then decrease the flow on this path by $\Delta^{\mathfrak{s},\mathfrak{t}}_{(u,v)}$. This reduces f_{viol} and we can just repeat this process until f_{viol} is equal to zero. Since the flow is initially integral, $\Delta^{\mathfrak{s},\mathfrak{t}}_{(u,v)}$ is always integer and the repaired flow remains integral.

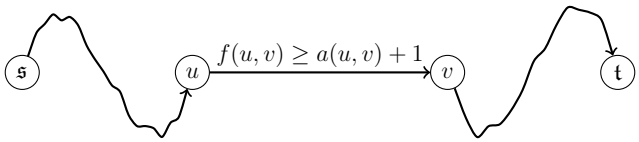


Figure 6.5: Illustration of a path with at least one unit flow from $\mathfrak{s}$ to $\mathfrak{t}$ through an arc (u, v) where the capacity is violated.

Step 3 – Maximum Flow. The last step is to run a maximum flow algorithm initialized with the possibly repaired flow. Any augmenting path algorithm can be used. For our implementation we used the algorithm described by Edmonds and Karp (1972).

6.5 Computational Results

We have compared our results to the best ones that can be found in the literature. For the ITC2007 competition, a benchmarking tool was provided. It returns the number of seconds that the competitors were allowed to run their algorithms on their own computers, restricted to one core. This number of seconds is referred to as one CPU unit. Our tests are conducted on an Intel® Core™ I5-3570K 3.4GHz CPU with 16GB RAM running Windows 10. The benchmark tool returned 208 seconds as one CPU unit for our computer. The ILP solver we used is from Gurobi Optimization Inc. (2015) version 6.5.

Since our mixed-integer programming (MIP) algorithm provides lower bounds, we are only comparing our algorithm with other MIP-based algorithms from the literature that obtain lower bounds. The algorithms we compare are those identified by LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Patterns (this paper). We have run the tests with time limits of 1, 10 and 40 CPU units as it has also been done for the other algorithms. The running time of the preprocessing and the generation of the valid inequalities is included in the total time spent by our algorithm. 21 data sets have been provided for the competition, however only 14 of these were available for some of the algorithms. All data sets can be obtained from Bonutti et al. (2016).

Table 6.2 – 6.4 presents results for comparing these different algorithms on the first fourteen data sets. For each algorithm, we provide the lower bound obtained (lb) and the relative distance ($\%gap$) to the best known upper bound (ub) calculated as $(ub - lb)/ub$. The lower bounds in bold font are best lower bounds. Those underlined are obtained by a single algorithm.
Line Avg. $\%gap$ specifies the average percentage gap obtained for all the fourteen instances. The two last lines report the number of times an algorithm obtains a best lower bound (Best lb) and a better one than the others (Better lb).

Table 6.2: Lower bounds for various algorithms given 1 CPU time unit.

		LL12		BMPR10		HB11		CCRT13		Patterns	
Instance	*ub*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*
comp01	5	4	20.0	0	100.0	4	20.0	**5**	0.0	0	100.0
comp02	24	0	100.0	0	100.0	**10**	58.3	0	100.0	**10**	58.3
comp03	64	0	100.0	25	60.9	26	59.4	24	62.5	**41**	35.9
comp04	35	22	37.1	**35**	0.0	**35**	0.0	**35**	0.0	**35**	0.0
comp05	284	92	67.6	119	58.1	19	93.3	6	97.9	**154**	45.8
comp06	27	7	74.1	13	51.9	12	55.6	0	100.0	**19**	29.6
comp07	6	0	100.0	**6**	0.0	5	16.7	0	100.0	**6**	0.0
comp08	37	30	18.9	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0
comp09	96	37	61.5	68	29.2	39	59.4	**92**	4.2	82	14.6
comp10	4	2	50.0	3	25.0	**4**	0.0	0	100.0	**4**	0.0
comp11	0	**0**	0.0	**0**	0.0	**0**	0.0	-	-	-	-
comp12	298	29	90.3	101	66.1	43	85.6	0	100.0	**109**	63.4
comp13	59	33	44.1	52	11.9	46	22.0	57	3.4	**59**	0.0
comp14	51	40	21.6	41	19.6	41	19.6	32	37.3	**45**	11.8
Avg. *%gap*			56.5		37.3		35.0		50.4		27.9
Best *lb*		**1**		**4**		**5**		**4**		**11**	
Better *lb*								**2**		**6**	

In Table 6.2, it can be seen that the pattern formulation yields a best lower bound for 11 of the 14 data sets. The three exceptions are for data sets comp01, comp09 and comp11. For data set comp01, we do not achieve the best lower bound because we do not consider the room penalties whereas the others do. For data set comp11, the number of time slots per day is 9, resulting in a potential of 512 patterns for each course and each day. In this case, our algorithm reaches the time limit of 1 CPU unit before completing the preprocessing and the valid inequality generation. This is also the case when we look at Table 6.3. In Table 6.4, it can be seen that our algorithm is the best for 13 instances, only producing a worse bound again for data set comp01. In half of the instances, the pattern formulation provides a better bound. Furthermore, for instance comp12, the lower 165 are an improvement over the currently best known bound 138.

Because Cacchiani et al. (2013) also report results on the last seven instances for a time limit of 40 CPU units, we also compare our algorithm to their algorithm on all data sets, except comp11. Indeed, an optimal solution of this data set has a zero cost and therefore Cacchiani et al. (2013) deemed this uninteresting as a trivial lower bound is also zero. The results are presented in Table 6.5 where it can be seen that the pattern formulation provides the best lower bound 19 times out of 20. We only get a worse bound for data set comp01 because we do not consider any room related penalties. Finally, in half of these data sets, a better lower bound is obtained by our algorithm compared to CCRT13. The last comparison is against the best known lower bounds as reported on the website of Bonutti et al. (2016). The algorithms or the computational times used for these bounds are not always clear, so we have chosen to run our algorithm with a time limit of 100 CPU units to see if we can improve some of the bounds. The results are presented in Table 6.6 where it can be seen that for comp03, comp12 and comp15, new lower bounds are obtained. The pattern formulation is also able to match

Table 6.3: Lower bounds for various algorithms given 10 CPU time units.

		LL12		BMPR10		HB11		CCRT13		Patterns	
Instance	*ub*	*lb*	%*gap*	*lb*	%*gap*	*lb*	%*gap*	*lb*	%*gap*	*lb*	%*gap*
comp01	5	4	20.0	4	20.0	4	20.0	**5**	0.0	0	100.0
comp02	24	8	66.7	0	100.0	12	50.0	**16**	33.3	14	41.7
comp03	64	0	100.0	33	48.4	34	46.9	**52**	18.8	46	28.1
comp04	35	28	20.0	**35**	0.0	**35**	0.0	**35**	0.0	**35**	0.0
comp05	284	25	91.2	111	60.9	69	75.7	6	97.9	**178**	37.3
comp06	27	10	63.0	15	44.4	12	55.6	11	59.3	**20**	25.9
comp07	6	2	66.7	**6**	0.0	**6**	0.0	**6**	0.0	**6**	0.0
comp08	37	34	8.1	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0
comp09	96	41	57.3	65	32.3	67	30.2	92	4.2	**94**	2.1
comp10	4	**4**	0.0	**4**	0.0	**4**	0.0	2	50.0	**4**	0.0
comp11	0	**0**	0.0	**0**	0.0	**0**	0.0	-	-	-	-
comp12	298	32	89.3	95	68.1	78	73.8	0	100.0	**159**	46.6
comp13	59	39	33.9	52	11.9	53	10.2	57	3.4	**59**	0.0
comp14	51	41	19.6	42	17.6	43	15.7	48	5.9	**51**	0.0
Avg. %*gap*			45.6		28.8		27.0		26.6		21.7
Best *lb*		**2**		**5**		**5**		**6**		**10**	
Better *lb*								**3**		**6**	

Table 6.4: Lower bounds for various algorithms given 40 CPU time unit.

		LL12		BMPR10		HB11		CCRT13		Patterns	
Instance	*ub*	*lb*	%*gap*	*lb*	%*gap*	*lb*	%*gap*	*lb*	%*gap*	*lb*	%*gap*
comp01	5	4	20.0	**5**	0.0	4	20.0	**5**	0.0	0	100.0
comp02	24	11	54.2	1	95.8	12	50.0	16	33.3	**20**	16.7
comp03	64	25	60.9	33	48.4	36	43.8	**52**	18.8	**52**	18.8
comp04	35	28	20.0	**35**	0.0	**35**	0.0	**35**	0.0	**35**	0.0
comp05	284	108	62.0	114	59.9	80	71.8	166	41.5	**191**	32.7
comp06	27	10	63.0	16	40.7	16	40.7	11	59.3	**24**	11.1
comp07	6	**6**	0.0	**6**	0.0	**6**	0.0	**6**	0.0	**6**	0.0
comp08	37	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0
comp09	96	46	52.1	66	31.3	67	30.2	92	4.2	**96**	0.0
comp10	4	**4**	0.0	**4**	0.0	**4**	0.0	2	50.0	**4**	0.0
comp11	0	**0**	0.0	**0**	0.0	**0**	0.0	-	-	**0**	0.0
comp12	298	53	82.2	95	68.1	84	71.8	100	66.4	**165**	44.6
comp13	59	41	30.5	54	8.5	55	6.8	57	3.4	**59**	0.0
comp14	51	46	9.8	42	17.6	43	15.7	48	5.9	**51**	0.0
Avg. %*gap*			32.8		26.5		25.1		20.2		16.0
Best *lb*		**4**		**6**		**5**		**6**		**13**	
Better *lb*										**7**	

Table 6.5: Cacchiani et al. (2013) vs. our algorithm, given 40 CPU units.

		CCRT13		Patterns	
Instance	*ub*	*lb*	%*gap*	*lb*	%*gap*
comp01	5	**5**	0.0	0	100.0
comp02	24	16	33.3	**20**	16.7
comp03	64	**52**	18.8	**52**	18.8
comp04	35	**35**	0.0	**35**	0.0
comp05	284	166	41.5	**191**	32.7
comp06	27	11	59.3	**24**	11.1
comp07	6	**6**	0.0	**6**	0.0
comp08	37	**37**	0.0	**37**	0.0
comp09	96	92	4.2	**96**	0.0
comp10	4	2	50.0	**4**	0.0
comp12	298	100	66.4	**165**	44.6
comp13	59	57	3.4	**59**	0.0
comp14	51	48	5.9	**51**	0.0
comp15	62	**52**	16.1	**52**	16.1
comp16	18	13	27.8	**18**	0.0
comp17	56	48	14.3	**52**	7.1
comp18	61	**52**	14.8	**52**	14.8
comp19	57	48	15.8	**57**	0.0
comp20	4	**4**	0.0	**4**	0.0
comp21	74	**68**	8.1	**68**	8.1
Avg. %*gap*			18.1		12.9
Best *lb*		**9**		**19**	
Better *lb*		**1**		**10**	

most of the currently best known bounds, that is, 16 times out of 21. For the data sets where it produces worse bounds, it can be seen that it is still very close except for data set comp18. The reason for this is hard to identify as it is not known what was the time limit when this bound was obtained.

6.6 Perspective

The pattern formulation improves three of the lower bounds for the ITC2007 data sets. It also has some other benefits such as the possibility of including specialized constraints to the daily time patterns. For example, if a course has two lectures in a given day, these should not be too far apart, or if a course has any lectures in a day, there should be a minimum or maximum number of lectures scheduled that day. These constraints could either be hard or soft and it can easily be checked whether the patterns are violating them. Such constraints can even be included non-linearly, e.g. if the distance between two lectures for some courses scheduled on the same day is defined as the number of empty time slots between them, then such distance could be penalized by squaring the number or by some other non-linear expression.

Table 6.6: Comparison with the best known bounds given 100 CPU units.

	Best known			Patterns	
Instance	*ub*	*lb*	%*gap*	*lb*	%*gap*
comp01	5	5	0.0	0	100.0
comp02	24	24	0.0	**24**	0.0
comp03	64	52	18.8	**54**	15.6
comp04	35	35	0.0	**35**	0.0
comp05	284	211	25.7	210	26.1
comp06	27	27	0.0	26	3.7
comp07	6	6	0.0	**6**	0.0
comp08	37	37	0.0	**37**	0.0
comp09	96	96	0.0	**96**	0.0
comp10	4	4	0.0	**4**	0.0
comp11	0	0	0.0	**0**	0.0
comp12	298	138	53.7	**175**	41.3
comp13	59	59	0.0	**59**	0.0
comp14	51	51	0.0	**51**	0.0
comp15	62	52	16.1	**54**	12.9
comp16	18	18	0.0	**18**	0.0
comp17	56	56	0.0	53	5.4
comp18	61	61	0.0	52	14.8
comp19	57	57	0.0	**57**	0.0
comp20	4	4	0.0	**4**	0.0
comp21	74	74	0.0	**74**	0.0
Avg. %*gap*			5.4		10.5
Best *lb*				**16**	
Better *lb*				**3**	

The pattern formulation could be extended to consider an entire week instead of only one day at a time. This however results in many more patterns, which might require the implementation of a Branch-and-Price algorithm. One drawback is that we loose the benefit of the cutting planes already available in a commercial code such as Gurobi. A shift to a Branch-and-Price framework such as SCIP (Achterberg, 2009) could be a way to get around this issue.

In this paper, we describe how a conflict graph on the patterns is constructed. This is used to statically generate clique cuts for the pattern formulation. It could be interesting to find out if more edges can be added to the graph. Furthermore instead of adding the cuts statically, another approach could be to use the graph to generate the clique cuts dynamically as in a Branch-and-Cut algorithm.

References

Achterberg, T. (2009). "SCIP: Solving constraint integer programs". In: *Mathematical Programming Computation* 1.1. http://mpc.zib.de/index.php/MPC/article/view/4, pp. 1–41.

Asín Achá, R. and Nieuwenhuis, R. (2012). "Curriculum-based course timetabling with SAT and MaxSAT". In: *Annals of Operations Research*, pp. 1–21.

Avella, P. and Vasil'ev, I. (2005). "A computational study of a cutting plane algorithm for university course timetabling". In: *Journal of Scheduling*. ISSN: 10946136. DOI: 10.1007/s10951-005-4780-1.

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). "An overview of curriculum-based course timetabling". In: *TOP*, pp. 1–37.

Bonutti, A., De Cesco, F., Di Gaspero, L., and Schaerf, A. (2012). "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results". In: *Annals of Operations Research* 194.1, pp. 59–70.

Bonutti, A., Gaspero, L. D., and Schaerf, A. (2016). *Curriculum-Based Course TimeTabling*. http://tabu.diegm.uniud.it/ctt/index.php[Retrieved 30/12-2016].

Bron, C. and Kerbosch, J. (1973). "Algorithm 457: Finding All Cliques of an Undirected Graph". In: *Commun. ACM* 16.9, pp. 575–577. ISSN: 0001-0782. DOI: 10.1145/362342.362367. URL: http://doi.acm.org/10.1145/362342.362367.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2008). "Penalising Patterns in Timetables: Novel Integer Programming Formulations". In: *Operations Research Proceedings 2007*. Ed. by J. Kalcsics and S. Nickel. Vol. 2007. Operations Research Proceedings. 10.1007/978-3-540-77903-2_63. Springer Berlin Heidelberg, pp. 409–414. ISBN: 978-3-540-77903-2.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010). "A supernodal formulation of vertex colouring with applications in course timetabling". In: *Annals of Operations Research* 179.1, pp. 105–130.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). "A branch-and-cut procedure for the Udine Course Timetabling problem". In: *Annals of Operations Research* 194.1, pp. 71–87.

Cacchiani, V., Caprara, A., Roberti, R., and Toth, P. (2013). "A new lower bound for curriculum-based course timetabling". In: *Computers and Operation Research* 40.10, pp. 2466–2477. DOI: 10.1016/j.cor.2013.02.010.

Di Gaspero, L., McCollum, B., and Schaerf, A. (2007). *The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)*. Tech. rep. School of Electronics, Electrical Engineering and Computer Science, Queenes University SARC Building, Belfast, United Kingdom.

Edmonds, J. and Karp, R. M. (1972). "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems". In: *J. ACM* 19.2, pp. 248–264. ISSN: 0004-5411. DOI: 10.1145/321694.321699. URL: http://doi.acm.org.globalproxy.cvt.dk/10.1145/321694.321699.

Glover, F. and Laguna, M. (2013). "Handbook of Combinatorial Optimization". In: ed. by M. P. Pardalos, D.-Z. Du, and L. R. Graham. New York, NY: Springer New York. Chap. Tabu Search, pp. 3261–3362. ISBN: 978-1-4419-7997-1. DOI: 10.1007/978-1-4419-7997-1_17. URL: http://dx.doi.org/10.1007/978-1-4419-7997-1%5C_17.

Gurobi Optimization Inc. (2015). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Hao, J. K. and Benlic, U. (2011). "Lower bounds for the ITC-2007 curriculum-based course timetabling problem". In: *European Journal of Operational Research* 212.3, pp. 464–472.

Kou, L. T., Stockmeyer, L. J., and Wong, C.-K. (1978). "Covering edges by cliques with regard to keyword conflicts and intersection graphs". In: *Communications of the ACM* 21.2, pp. 135–139.

Lach, G. and Lübbecke, M. (2012). "Curriculum based course timetabling: new solutions to Udine benchmark instances". In: *Annals of Operations Research* 194, pp. 255–272. ISSN: 0254-5330.

Lach, G. and Lübbecke, M. E. (2008). "Optimal University Course Timetables and the Partial Transversal Polytope". In: *International Workshop on Experimental and Efficient Algorithms*. Springer, pp. 235–248.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., and Burke, E. K. (2010). "Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition". In: *INFORMS Journal on Computing* 22.1, pp. 120–130.

Van Roy, T. J. and Wolsey, L. A. (1987). "Solving mixed integer programming problems using automatic reformulation". In: *Operations Research* 35.1, pp. 45–57.

7 Dantzig-Wolfe Decomposition of the Daily Course Pattern Formulation for Curriculum-based Course Timetabling

Niels-Christian F. Bagger[a,b] · Matias Sørensen[a,b] · Thomas R. Stidsen[a]

[a]mORetime research group, Management Science, Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 426B, DK-2800 Kgs. Lyngby, Denmark, http://www.moretime.man.dtu.dk/

[b]MaCom A/S, Vesterbrogade 48, 1., DK-1620 København V, Denmark

Status: Submitted to European Journal of Operational Research

Abstract: In this paper, we consider the Curriculum-based Course Timetabling problem, which consists of assigning weekly lectures to a time schedule and assign them to rooms. We develop a Column Generation algorithm based on a pattern formulation of the time scheduling part of the problem. The pattern formulation is an enumeration of all schedules to which each course can be assigned on each day, and it is a lower bounding model. We apply a Dantzig-Wolfe Decomposition that exploits the block diagonal structure of the pattern formulation such that we have a pricing problem for each day.

We provide a preprocessing technique that on average removes more than 40% of the pattern variables of the pricing problems. We then extend the preprocessing technique into inequalities that we add to the model. Lastly, we describe how we apply Local Branching to the pricing problem by using columns generated in previous iterations.

We compare the lower bounds we obtain with other methods from literature on 20 data instances originating from real-world applications. For 16 of the instances the optimal solutions are known, but the remaining four are still *open*. The average gap between the lower bounds obtained and the best-known solutions is lower than the other approaches from the literature, except the pattern formulation from which the decomposition originates from. However, our approach is able to improve the best-known lower bound for all four open instances, which decreases the average gap from 24% to 11%.

Keywords: Timetabling · Integer Programming · Education · Column Generation · Local Branching

7.1 Introduction

In this article we focus on the Curriculum-based Course Timetabling problem (CTT) as described by Di Gaspero et al. (2007). The problem has received much attention as it was used for the second International Timetabling Competition in 2007 (ITC2007) (Di Gaspero et al., 2007; McCollum et al., 2010). Following the competition, a website (The Scheduling and Timetabling Research Group at the University of Udine, Italy, 2015) has been created where researchers can upload data instances, solutions and bounds. Most of the work conducted on CTT is dominated by heuristic approaches (Asín Aschá and Nieuwenhuis, 2014; Lübbecke, 2015). These heuristic methods have provided most of the best-known solutions according to the website The Scheduling and Timetabling Research Group at the University of Udine, Italy (2015). For ITC2007 21 data instances were provided arising from real-world applications. Four of these instances are still *open*, meaning that the best-known upper bound does not equal the best-known lower bound. The lower bounds are necessary as the heuristics themselves do not provide a quality measurement. Our goal of this work is to strengthen the best-known lower bounds. We do this by applying a Dantzig-Wolfe decomposition of a previous formulation which is solved by column generation. Column generation approaches have been considered before by Cacchiani et al. (2013), but it is still worthwhile to investigate such methods further (Lübbecke, 2015).

In section 7.1.1 we describe the problem in details, and in section 7.1.2 we provide an overview of other methods in literature that have considered CTT. In section 7.2 we describe the pattern formulation suggested by Bagger et al. (2016) as this model is the basis of our work in this article. We assume that the reader is familiar with Dantzig-Wolfe Decomposition and Column Generation. However, in section 7.3 we provide a brief overview of the techniques followed by our application of the decomposition of the pattern formulation. In section 7.4 we describe the preprocessing techniques we apply, as well as some inequalities we derive followed by the framework we use in the solution process; Local Branching (Fischetti and Lodi, 2003). We report the results of our computational experiments in section 7.5, and lastly, we provide our conclusion in section 7.6.

7.1.1 Curriculum-based Course Timetabling

The CTT problem consists of the following entities; courses, days, time slots, lecturers, rooms and curricula. Each course is taught by exactly one lecturer and contains lectures that must all be scheduled in a weekly timetable and assigned to rooms. The week is divided into days, and each day is divided into time slots which are all equal in size. A day and time slot pair is referred to as a period, so the total number of periods is the number of days multiplied by the number of time slots. The length of one lecture corresponds to one period. A curriculum is a set of courses where for every pair there is a set of students attending both courses.

The hard constraints are as follows: All lectures of a course must be scheduled, and they must be scheduled in different periods. If a lecture is not scheduled, then it is counted as one violation of the Lectures (**L**) constraint, and if two lectures of the same course are scheduled in the same period, then this is also counted as one violation. A course can have specific periods

defined as unavailable periods. Every lecture scheduled in such a period is considered as one violation of the Availability (**A**) constraint. The constraint Conflicts (**C**) is violated by one if two courses that are taught by the same lecturer or belonging to the same curriculum have lectures scheduled in the same periods. Every room cannot accommodate more than one lecture in any given period. If more than one lecture is scheduled in the same room and same period, then the constraint Room Occupancy (**RO**) is violated by one for every lecture minus one.

Besides the hard constraints, the problem also contains four soft constraints. We are allowed to schedule any course into any room. However, it is desired to be able to accommodate as many students as possible when scheduling the courses into rooms. Every room has a capacity, and if the number of students attending a lecture is larger than the capacity of the room that the lecture is assigned, the constraint Room Capacity (**RC**) is violated by one for each student more than the capacity. Furthermore, as the courses contain multiple lectures, it can also be an advantage that the lectures are all scheduled in the same room during the week. For every course the constraint Room Stability (**RStab**) is violated by one for every distinct room to which the course is assigned minus one. For every course it is preferred to spread the lectures across a predetermined number of days. This number is called *minimum working days*. If the lectures are scheduled in fewer days than the minimum working days, then the constraint Minimum Working Days (**MWD**) is violated by one for each day below the minimum working days on which the lectures are scheduled. The last soft constraint is the Isolated Lectures (**IL**) constraint. If two periods belong to the same day and are in consecutive time slots, then we say that the periods are *adjacent*. Consider a curriculum and a course belonging to the curriculum. If the course has a lecture scheduled in a period and no lecture from any of the courses belonging to the curriculum has a lecture scheduled in an adjacent period, then we say that the lecture is *isolated*. For every curriculum, the constraint Isolated Lectures (**IL**) is violated by one for every isolated lecture.

Note that the **IL** constraint is usually referred to as the *curriculum compactness* constraint in literature. We use the name *isolated lectures* as Bonutti et al. (2012) mentions different ways of defining *curriculum compactness*, and they use the name *isolated lectures* for the formulation used here and in ITC2007.

Any feasible timetable must fulfil all the hard constraints, i.e., a timetable is considered feasible if, and only if, all the hard constraints have no violations. The objective is then to find a feasible timetable while minimising the soft constraints. Each soft constraint has a weight associated such that a single objective is defined by a weighted sum of all the soft constraints.

7.1.2 Related Work

In this section, we describe approaches from literature that have considered CTT. As our method is a lower bounding method, we focus on other lower bounding methods for CTT in literature. For a comprehensive overview of the literature regarding CTT, we refer to Bettinelli et al. (2015).

Burke et al. (2010a) introduce an exact mixed integer programming (MIP) model of CTT. They formulate the **IL** by using a variable for each curriculum and each period. Burke et al. (2008) remove those variables and instead they have just one variable for each curriculum and each day. The value of this variable is then calculated by adding exponentially many constraints. In Burke et al. (2012) they keep a subset of the beforementioned constraints and then add the

remaining dynamically whenever they are violated. Burke et al. (2010b) take the model from Burke et al. (2010a) and split it into two stages; first the courses are scheduled into periods and then they are assigned to rooms. This approach is executed iteratively.

Splitting the problem into two stages is also considered by Lach and Lübbecke (2008) and Lach and Lübbecke (2012), where the problem is also split into two stages; the first stage schedules the courses to periods and assign them to capacities, and the second stage then assigns the rooms with respect to the assigned capacities.

Hao and Benlic (2011) consider the first stage problem of Lach and Lübbecke (2012). They make a decomposition by relaxing some of the constraints such that the problem can be divided into subproblems. They then compute a lower bound for each subproblem and sum them up to get a lower bound for the overall problem.

Cacchiani et al. (2013) also compute lower bounds. They do this by splitting the problem into two parts, where one part considers the time related constraints and the other part considers the room related constraints. A lower bound is then calculated by summing up lower bounds for both parts. They, therefore, apply a Dantzig-Wolfe decomposition of the part with the time related constraints such that the pricing problem is decomposable by days and the model is solved by column generation.

Asín Aschá and Nieuwenhuis (2014) propose multiple satisfiability encodings. They start by treating the soft constraints as hard constraints and solve the problem as a pure satisfiability problem. Then they *relax* the constraints one by one, to move towards a weighted partial maximum satisfiability encoding.

In Bagger et al. (2017) a decomposition of the problem is considered similar to Lach and Lübbecke (2012) and Burke et al. (2010b) where the problem is split into a time scheduling model and a room allocation model. The two models are then reconnected by an underlying flow problem to get an exact formulation.

In Bagger et al. (2016) the time scheduling part of the problem is considered. Here a pattern formulation is suggested where each variable corresponds to a time schedule for one course on one day.

In this article we apply Dantzig-Wolfe decomposition to the pattern formulation in Bagger et al. (2016). The decomposition makes the pricing problem decomposable by days, and we solve the problem by a column generation algorithm. Note that this is similar to the work by Cacchiani et al. (2013). Cacchiani et al. (2013) apply the decomposition on the basis of a formulation where the main decision variables each represent one course and one period. Furthermore, the formulation does not guarantee a feasible schedule, i.e., some of the hard constraints can be violated. We apply the decomposition on the pattern formulation described in Bagger et al. (2016) which is an extensive model where each variable represents an entire schedule for one course on one day. Furthermore, the formulation ensures that none of the hard constraints is violated.

7.2 Pattern Formulation

In this section we provide an overview of the pattern formulation provided by Bagger et al. (2016), which we refer to for details. The idea of the pattern formulation is to have a binary variable represent an entire schedule for a course on one day. This formulation contains an exponential number of variables. However, as most of the data instances used in literature have

five or six time slots for each day, then the number of variables for each course and each day is 32 or 64. Before we describe the pattern formulation, we start by providing the notation used throughout this article. The set of courses, days and time slots are denoted as $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{T}$ respectively. The combination of a day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ is referred to as a period. For a time slot $t \in \mathcal{T}$ the time slot that is right before t is denoted by $t-1$ and the time slot right after t is denoted $t+1$. The set of curricula is denoted $\mathcal{Q}$, and for each curriculum $q \in \mathcal{Q}$, the set $\mathcal{C}_q \subseteq \mathcal{C}$ is the set of courses that belongs to the curriculum q. The last set is the set of *course cliques* Γ. The course cliques are derived from a graph constructed by creating a node for each course. If two courses are taught by the same lecturer or belong to the same curriculum, then their corresponding nodes are connected by an edge. Next, all maximal cliques are enumerated Bron and Kerbosch (see 1973), i.e., for each clique $\gamma \in \Gamma$, every node (course) is connected to all the other nodes (courses) in the clique. The set of courses corresponding to the nodes in the course clique $\gamma \in \Gamma$ is denoted $\mathcal{C}_\gamma$.

For each course $c \in \mathcal{C}$, the number of lectures to schedule is given by the parameter L_c, and the requested minimum number of working days is given by the parameter $D_c^{\min}$. For each curricula $q \in \mathcal{Q}$ the parameter L_q is the total number of lectures that must be scheduled for the courses $\mathcal{C}_q$, i.e., $L_q = \sum_{c \in \mathcal{C}_q} L_c$. Lastly, for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$ the parameter $F_{c,d,t}$ is one if the course is available in the corresponding period, otherwise it is zero. If time slot $t \in \mathcal{T}$ is the first time slot, then the parameter $F_{c,d,t-1}$ is defined as zero and likewise if t is the last time slot, the parameter $F_{c,d,t+1}$ is zero for each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$.

As all periods are uniform it is only necessary to generate the different patterns that are possible for the set of time slots $\mathcal{T}$ once and then apply them to each course and day. An example of all the patterns is illustrated in Table 7.1 when $|\mathcal{T}| = 4$.

Table 7.1: Illustration of all the patterns for $|\mathcal{T}| = 4$. Each column corresponds to a pattern, and each row corresponds to a time slot. The symbol "×" indicates whether or not a pattern schedules a lecture in the corresponding time slot.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		×				×	×	×				×	×	×		×
1			×			×			×	×		×	×		×	×
2				×			×		×		×	×		×	×	×
3					×			×		×	×		×	×	×	×

The set of all the patterns is denoted $\mathcal{K}$. For each pattern $k \in \mathcal{K}$ and time slot $t \in \mathcal{T}$ the parameter a_t^k is set to one if k contains a lecture in t. If $t \in \mathcal{T}$ is the first time slot, then a_{t-1}^k is defined as zero and likewise if t is the last time slot, then a_{t+1}^k is defined to be zero. The number of lectures contained in pattern $k \in \mathcal{K}$ is denoted L_k, i.e., $L_k = \sum_{t \in \mathcal{T}} a_t^k$. For each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ the set $\mathcal{K}_{c,d} \subseteq \mathcal{K}$ denotes the set of patterns that is feasible for c on day d. A pattern is feasible for a course $c \in \mathcal{C}$ and days $d \in \mathcal{D}$ if assigning course c to the pattern on day d does not schedule c in any unavailable periods. In Bagger et al. (2016) some preprocessing techniques are presented to decrease the sizes of the sets $\mathcal{K}_{c,d}$ and we refer to that article for details, as we do not go through these techniques.

Let $\lambda_{c,d}^k$ be a binary variable taking value one if course $c \in \mathcal{C}$ is assigned pattern $k \in \mathcal{K}_{c,d}$ for day $d \in \mathcal{D}$. The following constraints ensure that every course selects exactly one pattern

for each day, that all lectures are scheduled, and that no more than one lecture is scheduled in any room in any period:

$$\sum_{k \in \mathcal{K}_{c,d}} \lambda_{c,d}^k = 1, \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{7.1}$$

$$\sum_{d \in \mathcal{D}, k \in \mathcal{K}_{c,d}} L_k \lambda_{c,d}^k = L_c, \quad \forall c \in \mathcal{C} \tag{7.2}$$

$$\sum_{c \in \mathcal{C}, k \in \mathcal{K}_{c,d}} a_t^k \lambda_{c,d}^k \leq R, \quad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{7.3}$$

The constraints (7.1) – (7.3) ensure that the constraints **A**, **L** and **RO** are not violated. To model constraint **C** a *pattern conflict graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed. For every course $c \in \mathcal{C}$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$ there is a node $v_{c,d}^k \in \mathcal{V}$ corresponding to the variable $\lambda_{c,d}^k$. If course $c_1 \in \mathcal{C}$ chooses pattern $k_1 \in \mathcal{K}_{c_1,d_1}$ for day $d_1 \in \mathcal{D}$ and course $c_2 \in \mathcal{C}$ chooses pattern $k_2 \in \mathcal{K}_{c_2,d_2}$ for day $d_2 \in \mathcal{D}$ and this results in a conflict, then there is an edge $e \in \mathcal{E}$ between the two nodes $v_{c_1,d_1}^{k_1}$ and $v_{c_2,d_2}^{k_2}$. These conflicts contain the **C** constraints, and in Bagger et al. (2016) more conflicts are identified to add more edges to the graph. A clique in the graph is a subgraph of the graph such that every node in this subgraph is connected by an edge to every other node in the subgraph. Let Θ be a set of cliques such that for each edge $(u, v) \in \mathcal{E}$ both the nodes u and v are contained in at least one clique. We refer to these cliques as *pattern cliques* to be able to distinct them from the course cliques Γ. For each pattern clique $\theta \in \Theta$ in the graph let $\mathcal{V}_\theta$ be the set of nodes in the clique. Adding the following constraints ensure that the **C** constraints are not violated:

$$\sum_{v_{c,d}^k \in \mathcal{V}_\theta} \lambda_{c,d}^k \leq 1, \quad \forall \theta \in \Theta \tag{7.4}$$

Let w_c be an integer variable calculating how much the soft constraint **MWD** is violated. The value of these variables can be calculated by the following constraints:

$$\sum_{d \in \mathcal{D}, k \in \mathcal{K}_{c,d} : L_c \geq 1} \lambda_{c,d}^k + w_c \geq D_c^{\min}, \quad \forall c \in \mathcal{C} \tag{7.5}$$

To calculate the violation of the soft constraint **IL** the parameter $\overline{a}_t^k$ is defined for each pattern $k \in \mathcal{K}$ and time slot $t \in \mathcal{T}$:

$$\overline{a}_t^k := a_t^k - \max\left\{a_{t-1}^k, a_{t+1}^k\right\}, \quad k \in \mathcal{K}, t \in \mathcal{T} \tag{7.6}$$

The variable $s_{q,d,t}$ is introduced for each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$. The variable $s_{q,d,t}$ is a binary variable that takes value one if q has an isolated lecture in time slot t for day d:

$$\sum_{c \in \mathcal{C}_q, k \in \mathcal{K}_{c,d}} \overline{a}_t^k \lambda_{c,d}^k \leq s_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{7.7}$$

Let $W^{\mathbf{MWD}}$ and $W^{\mathbf{IL}}$ be the non-negative weights of the soft constraints **MWD** and **IL** respectively. Then the objective function to minimize can be formulated as follows:

$$\sum_{q\in\mathcal{Q},d\in\mathcal{D},t\in\mathcal{T}} W^{\mathbf{IL}} s_{q,d,t} + \sum_{c\in\mathcal{C}} W^{\mathbf{MWD}} w_c \tag{7.8}$$

It was shown by Bagger et al. (2016) that if $D_c^{\min} = L_c$ or $D_c^{\min} = 2$, then the following substitutions can be made:

$$w_c = \sum_{\substack{d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}:L_k\geq 2}} (L_k - 1)\,\lambda_{c,d}^k, \quad \forall c \in \mathcal{C} : D_c^{\min} = L_c \tag{7.9}$$

$$w_c = \sum_{\substack{d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}:L_k=L_c}} \lambda_{c,d}^k, \qquad \forall c \in \mathcal{C} : D_c^{\min} = 2 \tag{7.10}$$

Consider a curriculum $q \in \mathcal{Q}$, a day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$. Let the set of courses $\mathcal{C}_{q,d,t}$ be defined as follows:

$$\mathcal{C}_{q,d,t} := \left\{ c \in \mathcal{C}_q \;\middle|\; \sum_{t'\in\{t-1,t,t+1\}} F_{c,d,t'} \geq 1 \right\} \tag{7.11}$$

It was shown by Bagger et al. (2016) that if the number of courses in $\mathcal{C}_{q,d,t}$ is one, the following substitution can be made:

$$s_{q,d,t} = \sum_{\substack{c\in\mathcal{C}_q,k\in\mathcal{K}_{c,d}:\\ a_t^k=1\wedge a_{t-1}^k=a_{t+1}^k=0}} \lambda_{c,d}^k, \forall q \in \mathcal{Q}, d \in \mathcal{D}, \in \mathcal{T} : |\mathcal{C}_{q,d,t}| = 1 \tag{7.12}$$

Bagger et al. (2016) also describes valid inequalities that are added to the model. We do not go through them here, but we refer to that article for details. In this article, we do one more preprocessing that is not described in Bagger et al. (2016). We consider the model including all the valid inequalities, but excluding the w and s variables and their associated constraints, i.e., we only consider the feasibility part of the model. We then iterate through each variable $\lambda_{c,d}^k$ and set the lower bound to one. Then we solve the LP relaxation, and if it is infeasible, we remove the variable from the model. Otherwise, we change the lower bound of $\lambda_{c,d}^k$ back to zero.

7.3 Dantzig-Wolfe Decomposition

In this section we provide a brief description of Dantzig-Wolfe decomposition followed by our application. In section 7.3.1 an introduction to the Dantzig-Wolfe decomposition for Mixed Integer Programs (MIP) is given, and in section 7.3.2 we describe how we apply it to the pattern formulation of CTT.

7.3.1 Brief Introduction to Dantzig-Wolfe Decomposition

Our introduction to the Dantzig-Wolfe Decomposition is specific for MIP models and we refer to Martin (1999, chapter 11) and Desrosiers and Lübbecke (2010) for thorough and more general descriptions. We consider an MIP of the form:

$$\begin{aligned} \min \quad & c^\top x && (7.13)\\ \text{s.t.} \quad & Ax \geq b && (7.14)\\ & Bx \geq d && (7.15)\\ & x \in \mathbb{Z}^n && (7.16) \end{aligned}$$

where c, x, b and d are vectors and A and B are matrices. To get a lower bound of the model (7.13) – (7.16) the linear programming (LP) relaxation is solved. In Figure 7.1 an example of the solution space is illustrated.

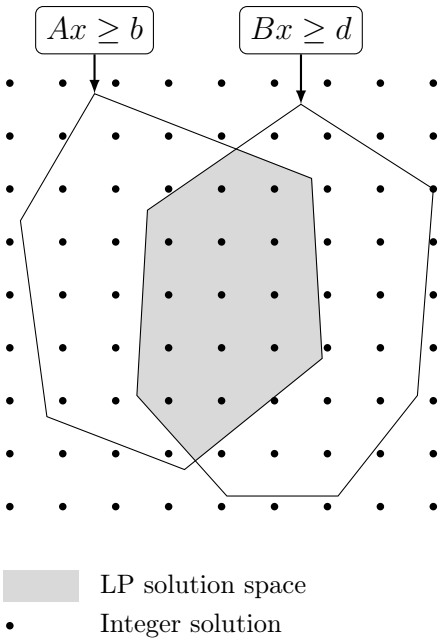


Figure 7.1: Illustration of the solution space.

The idea of the Dantzig-Wolfe decomposition for an MILP is to take the convex hull of $\{x \in X \mid Bx \geq d\}$, and replace it by variables. For simplicity, we assume that the convex hull $conv\,(\{x \in X \mid Bx \geq d\})$ is a polytope. Then any point $\overline{x}$ in this polytope can be written as a convex combination of the extreme points, i.e., if $\left\{\overline{x}^h\right\}_{h \in \mathcal{H}}$ is the set of all the extreme points of $conv\,(\{x \in \mathbb{R} \mid Bx \geq d\})$, then $\overline{x}$ can be written as follows:

$$\sum_{h\in\mathcal{H}} \overline{x}^h \lambda^h = \overline{x} \tag{7.17}$$

$$\sum_{h\in\mathcal{H}} \lambda^h = 1 \tag{7.18}$$

$$\lambda^h \geq 0, \quad \forall h \in \mathcal{H} \tag{7.19}$$

We can take this representation and insert it into the LP-relaxation of the model (7.13) – (7.16):

$$\min \quad \sum_{h\in\mathcal{H}} \left(c^\top \overline{x}^h\right) \lambda^h \tag{7.20}$$

$$\text{s.t.} \quad \sum_{h\in\mathcal{H}} \left(A\overline{x}^h\right) \lambda^h \geq b \tag{7.21}$$

$$\sum_{h\in\mathcal{H}} \lambda^h = 1 \tag{7.22}$$

$$\lambda^h \geq 0, \quad \forall h \in \mathcal{H} \tag{7.23}$$

Model (7.20) – (7.23) is referred to as the LP relaxation of the Dantzig-Wolfe *master problem*. Let z^*_{IP} be the optimal objective value of the model (7.13) – (7.16), let z^*_{LP} be the optimal objective value of the LP relaxation of the same model and let z^*_{DW} be the optimal objective value of the model (7.20) – (7.23). Then the benefit of rewriting the model is that the LP relaxation of the Dantzig-Wolfe master problem is a stronger relaxation in the sense that we have the following relation:

$$z^*_{LP} \leq z^*_{DW} \leq z^*_{IP} \tag{7.24}$$

Figure 7.2 illustrates the impact on the solution space when $Bx \geq d$ from Figure 7.1 is replaced by $conv\left(\{Bx \geq d, x \in \mathcal{Z}^n\}\right)$.

Explicitly describing model (7.20) – (7.23) can be difficult as the number of extreme points in $conv\left(\{x \in X \mid Bx \geq d\}\right)$ can be exponentially large. So a way to solve the model is to start with a restricted set $\mathcal{H}' \subseteq \mathcal{H}$ and then solve the model (7.20) – (7.23) with this restricted set. The model with this restricted set is referred to as the *restricted master problem* (RMP). Let π be the dual vector of the constraints (7.21) and π_0 be the dual variable of constraint (7.22). For a dual solution $(\overline{\pi}, \overline{\pi}_0)$ the reduced cost of a column $h \in \mathcal{H}$ is $\overline{c}^h = (c - A^\top \overline{\pi})^\top \overline{x}^h - \overline{\pi}_0$. If the dual solution is optimal then the RMP is optimal when $\overline{c}^h \geq 0$ for every $h \in \mathcal{H}$. We know that $\overline{c}^h \geq 0$ for every $h \in \mathcal{H}'$, but there may exist some column $h \in \mathcal{H}\backslash\mathcal{H}'$ where $\overline{c}^h < 0$ so we need to check if any such column exist. This can be done by solving the following problem, known as the *pricing problem* (PP):

$$\min \quad (c - A^\top \overline{\pi})^\top x - \overline{\pi}_0 \tag{7.25}$$

$$\text{s.t.} \quad Bx \geq d \tag{7.26}$$

$$x \in \mathbb{Z}^n \tag{7.27}$$

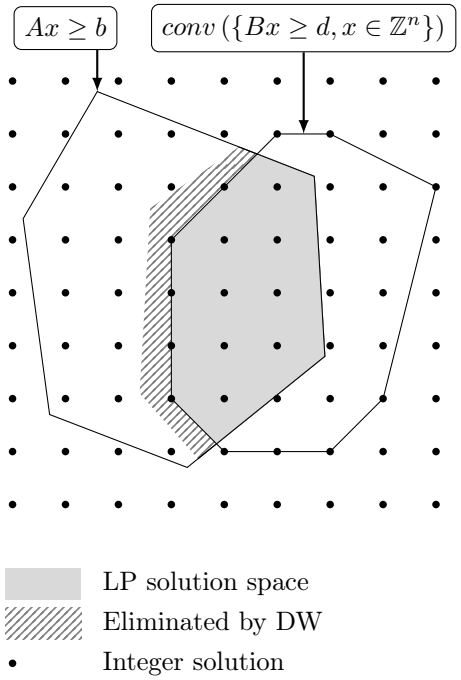


Figure 7.2: Illustration of the solution space of the master problem.

If PP contains any solution where the objective value is negative, then the RMP is not proven optimal, and we need to add the solution as a column to the RMP. This process is known as the *column generation* algorithm. First solve the RMP to obtain the dual solution $(\overline{\pi}, \overline{\pi}_0)$. Given the dual solution find a solution for model (7.25) – (7.27) with a negative objective value. If a solution with a negative reduced cost exists, then extend the restricted set $\mathcal{H}'$ with this solution and iterate the process. We continue this iterative process until the model (7.25) – (7.27) does not contain any solution with a negative objective value. This iterative process is illustrated in Figure 7.3.

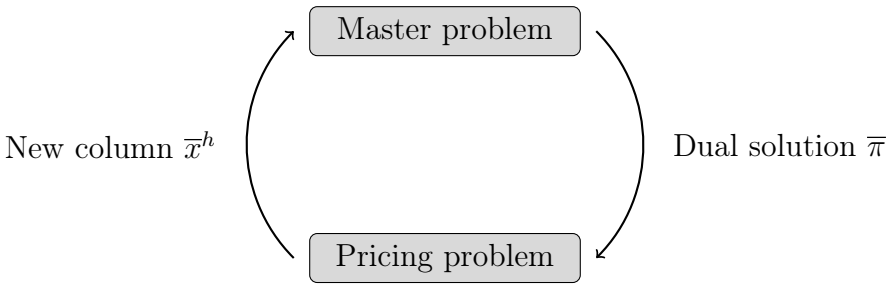


Figure 7.3: The iterative loop of the column generation algorithm.

Another benefit of the Dantzig-Wolfe decomposition is that it is possible to exploit if the constraint matrix B has a *block diagonal* structure as follows:

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_m \end{bmatrix}, \quad d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix} \tag{7.28}$$

Let $\mathcal{I} = \{1, 2, \ldots, m\}$ and for each $i \in \mathcal{I}$ let $\mathcal{H}_i$ be the extreme points of $conv\left(\{x_i \in \mathbb{Z} \,|\, B_i x_i \geq d_i\}\right)$ where x_i is the subset of variables in x that corresponds to the submatrix B_i. Similarly, c_i and A_i are the subvector and submatrix of the vector c and matrix A corresponding to the submatrix B_i. Then the LP relaxation of the master problem can be written as follows:

$$\min \quad \sum_{i \in \mathcal{I}, h \in \mathcal{H}_i} \left(c_i^\top \overline{x}_i^h\right) \lambda_i^h \tag{7.29}$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{I}, h \in \mathcal{H}_i} \left(A_i \overline{x}_i^h\right) \lambda_i^h \geq b \tag{7.30}$$

$$\sum_{h \in \mathcal{H}_i} \lambda_i^h = 1, \quad \forall i \in \mathcal{I} \tag{7.31}$$

$$\lambda_i^h \geq 0, \quad \forall i \in \mathcal{I}, h \in \mathcal{H}_i \tag{7.32}$$

The columns with a negative reduced cost are then found by solving the m independent pricing problems. We use this idea to decompose the CTT.

7.3.2 Dantzig-Wolfe for the Pattern Formulation

Here, we describe how we apply the Dantzig-Wolfe decomposition to the CTT problem by using the pattern formulation described in section 7.2. In the introduction of Martin (1999, chapter 11) it is mentioned that the decomposition should be chosen such that the pricing problem contains a *vast majority* of the constraints and such that the pricing problem has a *special* structure. The model we decompose is the model from section 7.2 with the additional preprocessing that we mentioned at the end of the section. However, we do not include all the valid inequalities described by Bagger et al. (2016). We only include the ones where all the variables that contribute can be associated with a single day. The reason is that we decompose the model such that we have a pricing problem for each day $d \in \mathcal{D}$ and we want to keep the master problem simple. So we only keep the constraints that ensure integer feasibility or where the variables contained correspond to the same day. Figure 7.4 illustrates the constraint matrix of the model applied to data instance comp03 from the ITC2007 competition. In the figure, the black pixels correspond to non-zero entries in the matrix. The large rectangles illustrate the separation of the A and B matrix and the block diagonal structure in B after we have sorted the variables and constraints according to the days.

In Figure 7.4 the top rectangle is the A matrix and the rectangles below correspond to the B matrix, where each rectangle is associated with a day. So decomposing the model according to days exploits the block diagonal structure and put a *vast majority* of the constraints in the pricing problems.

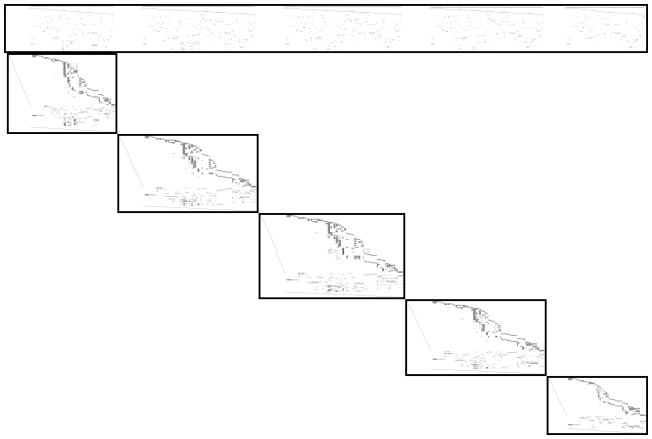


Figure 7.4: Illustration of the block diagonal structure of the constraint matrix for data instance comp03 when ordered by days.

For each day $d \in \mathcal{D}$ let $\mathcal{H}_d$ be the set of columns associated with d. Let x_b^h be a binary variable that takes value one if column $h \in \mathcal{H}_d$ is selected for day $d \in \mathcal{D}$ and let α_d^h be the associated cost. For each day $d \in \mathcal{D}$ and column $h \in \mathcal{H}_d$ the parameter $\overline{\lambda}_{c,d}^{k,h}$ is one if the column assigns course $c \in \mathcal{C}$ to pattern $k \in \mathcal{K}_{c,d}$. Furthermore, we define the set $\mathcal{C}_{\min} \subseteq \mathcal{C}$ as the set of courses where the w substitutions (7.9) and (7.10) do not apply. Lastly, we define the set $\Theta_{\geq 2} \subseteq \Theta$, which is the set of the pattern cliques in Θ that contains variables from at least two different days. We formulate the LP relaxation of our master problem as follows:

$$\min \sum_{c \in \mathcal{C}_{\min}} W^{\mathbf{MWD}} w_c + \sum_{d \in \mathcal{D}, h \in \mathcal{H}_d} \alpha_d^h x_d^h \tag{7.33}$$

$$\text{s.t.} \sum_{\substack{d \in \mathcal{D}, h \in \mathcal{H}_d, \\ k \in \mathcal{K}_{c,d}}} L_k \overline{\lambda}_{c,d}^{k,h} x_d^h = L_c, \quad \forall c \in \mathcal{C} \tag{7.34}$$

$$\sum_{\substack{d \in \mathcal{D}, h \in \mathcal{H}_d, \\ k \in \mathcal{K}_{c,d}: L_k \geq 1}} \overline{\lambda}_{c,d}^{k,h} x_d^h + w_c \geq D_c^{\min}, \quad \forall c \in \mathcal{C}_{\min} \tag{7.35}$$

$$\sum_{h \in \mathcal{H}_d, c \in \mathcal{C}, k \in \mathcal{K}_{c,d}} a_t^k \overline{\lambda}_{c,d}^{k,h} x_d^h \leq R, \quad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{7.36}$$

$$\sum_{v_{c,d}^k \in \mathcal{V}_\theta, h \in \mathcal{H}_d} \overline{\lambda}_{c,d}^{k,h} x_d^h \leq 1, \quad \forall \theta \in \Theta_{\geq 2} \tag{7.37}$$

$$\sum_{h \in \mathcal{H}_d} x_d^h \leq 1, \quad \forall d \in \mathcal{D} \tag{7.38}$$

$$x_d^h \geq 0, \quad \forall d \in \mathcal{D}, h \in \mathcal{H}_d \tag{7.39}$$

The model (7.33) – (7.39) is very similar to the master problem that Cacchiani et al. (2013) use for their column generation algorithm. The main difference is that we use the pattern formulation instead of a compact formulation and we also include the **RO** constraints (7.36). Note that we have replaced the equality sign in the convexity constraint (7.2) with an inequality

sign. The reason is that if we consider a column for a day where no lecture is scheduled, then this column has a cost of zero, and it will not contribute to any other constraint. We solve the model by the beforementioned column generation algorithm. The first columns we add are found by solving the model described in section 7.2 excluding the variables w and s and all the associated constraints to these variables, i.e., we consider only the feasibility part. The solution to the model is then the first set of columns.

Let the dual variables of the constraints (7.34), (7.35), (7.36), (7.37) and (7.38) be denoted β_c, ϕ_c, $\mu_{d,t}$, ζ_θ and π^0_d respectively. Consider an optimal dual solution $\left(\overline{\beta}, \overline{\phi}, \overline{\mu}, \overline{\zeta}, \overline{\pi}^0\right)$ for the restricted master problem in some iteration of the column generation algorithm. We define the parameter $\overline{\phi}^k_c$ to be equal to $\overline{\phi}_c$ if $c \in \mathcal{C}_{\min}$ and $L_k \geq 1$, otherwise we set it to zero. For each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$ we define $\overline{\pi}^k_{c,d}$:

$$\overline{\pi}^k_{c,d} := L_k \overline{\beta}_c + \overline{\phi}^k_c + \sum_{t \in \mathcal{T}} a^k_t \overline{\mu}_{d,t} + \sum_{\theta \in \Theta_{\geq 2} : v^k_{c,d} \in \mathcal{V}_\theta} \overline{\zeta}_\theta \tag{7.40}$$

For a day $d \in \mathcal{D}$ we describe the pricing problem for that day in the following. let λ^k_c be a binary variable taking value one if course $c \in \mathcal{C}$ is assigned to pattern $k \in \mathcal{K}_{c,d}$ and zero otherwise. Let $s_{q,t}$ be a binary variable taking value one if curriculum $q \in \mathcal{Q}$ has an isolated lecture scheduled in time slot $t \in \mathcal{T}$. For each day $d \in \mathcal{D}$ let Θ_d be a set of cliques found in the same way as the cliques Θ in section 7.2. However, here we restrict the cliques to be in the subgraph of the pattern clique graph induced by considering only the nodes belonging to day d. Note that, as we apply the substitutions mentioned in section 7.2, the variable $s_{q,t}$ is only defined for $|\mathcal{C}_{q,d,t}| > 1$. Lastly, let $\alpha^k_{c,d}$ be the cost of pattern $k \in \mathcal{K}_{c,d}$ for course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ after the substitutions. We can then formulate the pricing problem for day $d \in \mathcal{D}$:

$$\min \sum_{\substack{q \in \mathcal{Q}, t \in \mathcal{T}: \\ |\mathcal{C}_{q,d,t}| > 1}} W^{\mathbf{IL}} s_{q,t} + \sum_{c \in \mathcal{C}, k \in \mathcal{K}_{c,d}} \left(\alpha^k_{c,d} - \overline{\pi}^k_{c,d}\right) \lambda^k_c - \overline{\pi}^0_d \tag{7.41}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}_{c,d}} \lambda^k_c = 1, \quad \forall c \in \mathcal{C} \tag{7.42}$$

$$\sum_{v^k_{c,d} \in \mathcal{V}_\theta} \lambda^k_c \leq 1, \quad \forall \theta \in \Theta_d \tag{7.43}$$

$$\sum_{c \in \mathcal{C}_q, k \in \mathcal{K}} \overline{a}^k_t \lambda^k_c \leq s_{q,t}, \quad \forall q \in \mathcal{Q}, t \in \mathcal{T} : |\mathcal{C}_{q,d,t}| > 1 \tag{7.44}$$

$$\lambda^k_c \in \mathbb{B}, \quad \forall c \in \mathcal{C}, k \in \mathcal{K}_{c,d} \tag{7.45}$$

$$s_{q,t} \in \mathbb{B}, \quad \forall q \in \mathcal{Q}, t \in \mathcal{T} \tag{7.46}$$

Consider a solution $\left(\overline{\lambda}, \overline{s}\right)$ to the pricing problem (7.41) – (7.46) for day $d \in \mathcal{D}$. If the solution has a negative objective value then we can add it to the master problem as a new column $h \in \mathcal{H}_d$ by setting $\overline{\lambda}^{k,h}_{c,d} = \overline{\lambda}^k_c$ for every course $c \in \mathcal{C}$ and pattern $k \in \mathcal{K}_{c,d}$.

We also include the valid inequalities from Bagger et al. (2016) that can be associated with the specific day. The constraints (7.36) could be included in the pricing problem instead of the master problem as each of them can be associated with a specific day. However, keeping these constraints in the master ensures that the pricing problem has a *special* structure. We describe how we can exploit this *special* structure in section 7.4.

7.4 Preprocessing, Inequalities and Solution Method for the Pricing Problem

The pricing problem described in section 7.3.2 can be hard to solve for a generic MIP solver. In this section we describe the techniques we use to speed up the solution process. In section 7.4.1 we describe how we remove some of the variables of the pricing problem in an iteration of the column generation algorithm. In section 7.4.2 we describe how we can use the presolving technique to derive inequalities. In section 7.4.3 we describe how we use Local Branching as described by Fischetti and Lodi (2003) to solve the pricing problem.

7.4.1 Preprocessing

In this section we provide a preprocessing technique to eliminate some of the variables from the pricing problem. The technique is based on the objective function coefficients of the λ variables. Since the coefficients change in each iteration of the column generation algorithm, the variables we remove in one iteration must be reinserted for the next iteration.

Consider some course $c \in \mathcal{C}$ in the pricing problem for the day $d \in \mathcal{D}$ in any iteration of the column generation algorithm. Consider the patterns $k_1, k_2 \in \mathcal{K}_{c,d}$ where $k_1 \neq k_2$. The idea of the preprocessing technique is to check for any feasible solution where c is assigned to k_2, whether assigning c to k_1 instead is still a feasible solution and whether the objective value does not increase by this change. First, we must be able to guarantee that the new solution is feasible. Here we exploit that we make the room occupancy constraints part of the master problem, so we only have to consider the constraints (7.42) and (7.43) in the pricing problem for feasibility. All the other constraints in the pricing problem are used for calculating the values of the s variables. Since c is assigned to k_2, then the value of the variable $\lambda_c^{k_2}$ is one. Assigning c to k_1 instead of k_2 corresponds to setting the value of the $\lambda_c^{k_2}$ to zero and the value of $\lambda_c^{k_1}$ to one. Both $\lambda_c^{k_1}$ and $\lambda_c^{k_2}$ are in the constraint (7.42) associated with c, which means that since the solution was feasible before, then this constraint cannot be violated in the new solution. Let $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d) \subseteq \mathcal{G}$ be the subgraph, where $\mathcal{V}_d \subseteq \mathcal{V}$ is the set of nodes associated with day d and $\mathcal{E}_d \subseteq \mathcal{E}$ is the set of edges where both end points are in $\mathcal{V}_d$. Every edge in $\mathcal{E}_d$ is contained in at least one of the constraints (7.43). To check if the constraints (7.43) are fulfilled we need to consider the neighbourhoods in $\mathcal{G}_d$ of the nodes $v_{c,d}^{k_1}, v_{c,d}^{k_2} \in \mathcal{V}_d$. Let the neighbourhood of $v_{c,d}^{k_1}$, excluding every node that corresponds to c, be denoted $\mathcal{N}_{c,d}^{k_1} \subseteq \mathcal{V}_d$. Note that $v_{c,d}^{k_1}$ and $v_{c,d}^{k_2}$ must be connected by an edge since c cannot be assigned to more than one pattern. Assume that every node in $\mathcal{N}_{c,d}^{k_1}$ is also a neighbour of $v_{c,d}^{k_2}$. $v_{c,d}^{k_2}$ may have other neighbours that are not neighbours of $v_{c,d}^{k_1}$. This case is illustrated in Figure 7.5.

For any solution where c is assigned to k_2 the values of all the variables that correspond to the neighbours of $v_{c,d}^{k_2}$ must be zero. Since $\mathcal{N}_{c,d}^{k_1}$ is contained in the neighbourhood of $v_{c,d}^{k_2}$, then all the variables in $\mathcal{N}_{c,d}^{k_1}$ must also be zero. When we reassign c from k_2 to k_1 it means that we are changing the value for $\lambda_c^{k_2}$ from one to zero and the value for $\lambda_c^{k_1}$ from zero to one. As all of the variables in $\mathcal{N}_{c,d}^{k_1}$ are also zero and we have now changed $\lambda_c^{k_2}$ to zero $\lambda_c^{k_1}$, then in all the constraints (7.43) where $\lambda_c^{k_1}$ contributes all other variables must be zero, which means that the solution must be feasible. We did not account for the nodes $v_{c,d}^{k}$ for $k \in \mathcal{K}_{c,d} \backslash \{k_1, k_2\}$, but as c is assigned to exactly one pattern, then all these variables must be zero as well. As the new

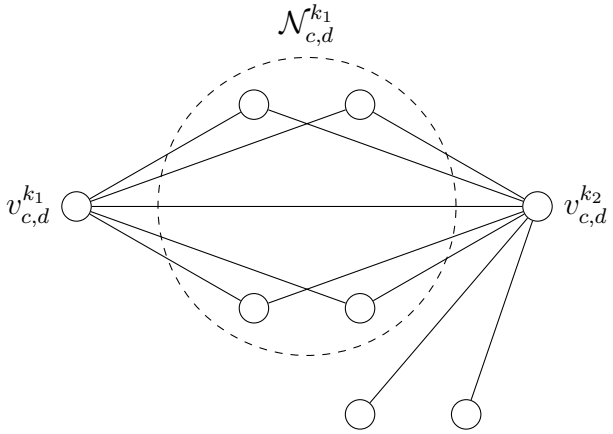


Figure 7.5: Illustration of the nodes in the pattern conflict graph $\mathcal{G}_d$ corresponding to the patterns; k_1 and k_2 for course c and the neighbourhood $\mathcal{N}_{c,d}^{k_1} \subseteq \mathcal{V}_d$.

solution is feasible, we now need to check if we can guarantee that the objective value does not increase.

Consider the objective function (7.41) which consists of a sum of the s variables followed by a sum of the λ variables and a constant. The change of the sum of the λ variables is the difference between the coefficients of $\lambda_c^{k_1}$ and $\lambda_c^{k_2}$ no matter how all the other courses are assigned. We cannot know the change in the sum of the s variables as it depends on the assigned patterns of the other courses. However, if we assume that we know an upper bound $\delta_{c,d}^{k_1,k_2}$ on how much the value can increase, then the total objective value cannot increase under the following condition:

$$\left(\alpha_{c,d}^{k_2} - \overline{\pi}_{c,d}^{k_2}\right) - \left(\alpha_{c,d}^{k_1} - \overline{\pi}_{c,d}^{k_1}\right) \geq \delta_{c,d}^{k_1,k_2} \tag{7.47}$$

The upper bound $\delta_{c,d}^{k_1,k_2}$ is determined by how many isolated lectures that at most get introduced when we make the reassignment. So we need to consider the difference in the time slots that are contained in the two patterns. As an example let $|\mathcal{T}| = 6$ and let pattern k_2 contain lectures in time slots t_2 and t_3 and let pattern k_1 contain lectures in time slots t_2, t_3 and t_5. This example is illustrated in a matrix in Figure 7.6. Each row in the matrix corresponds to a pattern and the columns correspond to the time slots. A cross denotes that the pattern contains a lecture in the corresponding time slot.

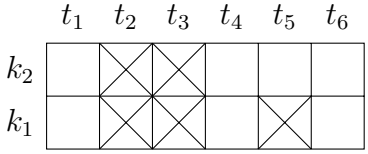


Figure 7.6: Illustration of the example of changing out pattern k_2 with k_1. Here a lecture is added.

As we are reassigning c from k_2 to k_1, then this means that c is assigned an extra lecture in time slot t_5. Since the pattern k_1 does not contain a lecture in either time slot t_4 nor in time slot t_6, then the lecture in t_5 is potentially a new isolated lecture for every curriculum that c

belongs to. So for every lecture that is added in the reassignment, if k_1 does not have a lecture in an adjacent time slot, then there is a potentially isolated lecture.

The next step is to consider the case where lectures are removed. Consider an example where k_2 contains lectures in time slots t_3 and t_5 and k_1 contains a lecture in t_3. This example is illustrated in Figure 7.7.

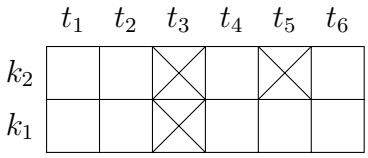


Figure 7.7: Illustration of the example of changing out pattern k_2 with k_1. Here a lecture is removed.

Before the reassignment, c has a lecture in t_5, and after the reassignment, it does not. The time slots that are adjacent, t_4 and t_6, can then become potentially isolated lectures as we do not know if some other courses belonging to the same curricula have lectures scheduled in these time slots. However as k_1 contains a lecture that is adjacent to t_4, then only t_6 can become a potential isolated lecture. Even though k_1 contains a lecture in t_3 and none in the adjacent time slots, then this is not counted as a potentially isolated lecture. The reason is that k_2 also contains a lecture in t_3, and therefore, if it is an isolated lecture, it would also have been so before, and thus does not change the objective value. So when a lecture gets removed from the reassignment, we consider the adjacent time slots as potentially isolated lectures unless there are lectures adjacent to those time slots in the new pattern.

After we have found all the potentially isolated lectures, we iterate through every curriculum that c belongs to, i.e., $\mathcal{Q}_c$. For each $q \in \mathcal{Q}_c$ we consider all the potentially isolated lectures previously found. We then remove every time slot t where $a_t^k = 0$ for every $k \in \mathcal{K}_{c',d}$ and every $c' \in \mathcal{C}_q$, i.e., if no course can be scheduled in t since this means that there cannot be an isolated lecture. As an example let the potential isolated lectures for q be in time slots t_2, t_3 and t_5 after the removal. This example is illustrated in Figure 7.8.

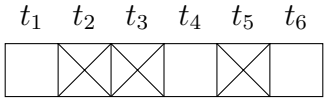


Figure 7.8: Illustration of the potential isolated lectures.

Let a sequence of these potentially isolated lectures be a set of consecutive time slots where there is a lecture in each of them but no lecture after the last and before the first lecture. In the example in Figure 7.8 there are two sequences; the first sequence consists of time slots t_2 and t_3, and the second sequence consists of time slot t_5. It is not possible to have two isolated lectures in adjacent time slots, so the maximum number of isolated lectures in each sequence is the number of time slots in the sequence divided by two and rounded up.

So to calculate the value of $\delta_{c,d}^{k_1,k_2}$ we start by setting it to zero. Then we iterate over all the curricula $\mathcal{Q}_c$. For each curriculum we iterate over every sequence $\{t_i, t_{i+1}, \ldots, t_j\}$, after we removed some of the time slots as mentioned before, and then we add the cost of the maximum number of isolated lectures in this sequence to $\delta_{c,d}^{k_1,k_2}$:

$$\delta_{c,d}^{k_1,k_2} \leftarrow \delta_{c,d}^{k_1,k_2} + W^{\mathbf{IL}} \left\lceil \frac{j-i+1}{2} \right\rceil \tag{7.48}$$

Now we have calculated an upper bound on the increase in the cost of the isolated lectures. For a course $c \in \mathcal{C}$ and pattern $k_1, k_2 \in \mathcal{K}_{c,d}$ we say that k_1 *dominates* k_2 if (7.47) is fulfilled. If one pattern dominates another, then it implies that we can remove the dominated pattern from the solution. However, we cannot remove all patterns that are dominated. As an example, consider a case where course c has five patterns; k_1, k_2, k_3, k_4 and k_5. Assume that k_1 dominates k_2, k_2 dominates k_3, k_3 dominates k_4, k_4 dominates k_5 and k_5 dominates k_1. This example is illustrated in Figure 7.9.

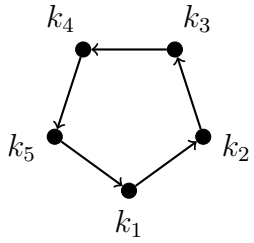


Figure 7.9: Illustration of five patterns of the example. The arrows illustrate the dominance between the patterns.

In the example, all the patterns are dominated so we could fix them all to zero. However, this would make the problem infeasible as c must select exactly one of them. So the way we do the preprocessing is by considering one pattern k at a time, and if it is dominated by another pattern which is not fixed to zero, then we fix k to zero, i.e., we change the upper bound of the variable from one to zero. In the example from Figure 7.9 we would for instance start with k_5. Since k_5 is dominated by k_4 and k_4 has not been fixed to zero, then we fix k_5 to zero. We then go to k_4 which is dominated by k_3, so we fix k_3 to zero and so we continue until we get to k_1. k_1 is dominated by k_5 but as k_5 has already been fixed to zero we do not fix k_1 to zero.

We apply the preprocessing technique described in this section before we solve the pricing problem in every iteration of the column generation algorithm. When we have solved the pricing problem we then *unfix* the variables again, i.e., we change the upper bounds of the variables back to one, so they are ready for the next iteration.

7.4.2 Optimality Inequalities

In this section we describe an extension to the preprocessing technique from section 7.4.1. Like in the preprocessing phase, the inequalities we derive here are only applicable in a single iteration of the column generation algorithm. Consider some course $c \in \mathcal{C}$ in the pricing problem for a day $d \in \mathcal{D}$ in any iteration of the column generation algorithm. Consider the patterns $k_1, k_2 \in \mathcal{K}_{c,d}$, where $k_1 \neq k_2$. In section 7.4.1 we only considered the patterns k_1 and k_2 where we could guarantee that if we had a feasible solution where c was assigned to k_2, then we could create a new feasible solution by reassigning c to k_1. In this section we do not keep this restriction, but consider every pair of patterns k_1 and k_2 where the condition (7.47) is fulfilled, i.e., where k_1 dominates k_2.

Let the neighbourhood of the node $v_{c,d}^{k_1}$ in $\mathcal{G}_d$, except for the nodes corresponding to c, be denoted $\mathcal{N}_{c,d}^{k_1} \in \mathcal{V}_d$. In section 7.4.1 every node in $\mathcal{N}_{c,d}^{k_1}$ was a neighbour of $v_{c,d}^{k_2}$. As we have removed this restriction in this section, then there might be some nodes in $\mathcal{N}_{c,d}^{k_1}$ which are not neighbours of $v_{c,d}^{k_2}$. We denote these nodes by $\mathcal{N}_{c,d}^{k_1 \setminus k_2}$, i.e., every node in $\mathcal{N}_{c,d}^{k_1 \setminus k_2}$ is not a neighbour of $v_{c,d}^{k_2}$, but every node in $\mathcal{N}_{c,d}^{k_1} \cap \mathcal{N}_{c,d}^{k_1 \setminus k_2}$ is a neighbour of $v_{c,d}^{k_2}$. An illustration of this is given in Figure 7.10.

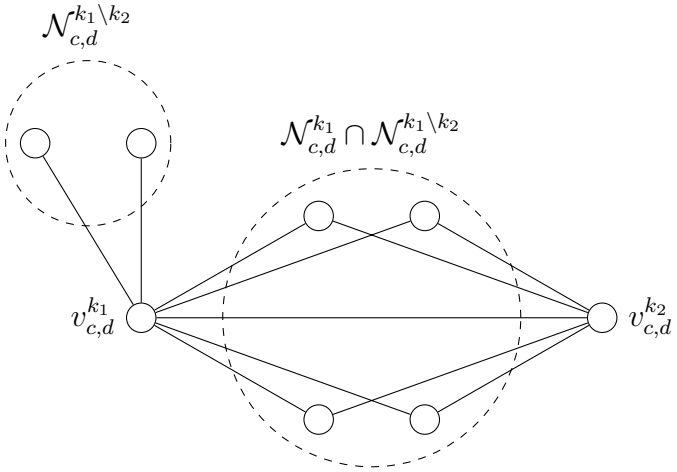


Figure 7.10: Illustration of the nodes in the pattern conflict graph $\mathcal{G}_d$ corresponding to the patterns; k_1 and k_2 for course c and the set of nodes $\mathcal{N}_{c,d}^{k_1 \setminus k_2} \subseteq \mathcal{V}_d$ and $\mathcal{N}_{c,d}^{k_1} \cap \mathcal{N}_{c,d}^{k_1 \setminus k_2}$.

Given any feasible solution where c is assigned to k_2 then reassigning c to k_1 is a feasible solution only if the values of the variables corresponding to the nodes in $\mathcal{N}_{c,d}^{k_1 \setminus k_2}$ are all zero. This means that if none of the variable in $\mathcal{N}_{c,d}^{k_1 \setminus k_2}$ is selected (has a value of one), then we can reassign c to k_1 without increasing the objective value. So it can only be beneficial to assign c to k_2 if at least one of the variables in $\mathcal{N}_{c,d}^{k_1 \setminus k_2}$ is selected, which leads us to the following inequality:

$$\lambda_c^{k_2} \leq \sum_{v_{c',d}^{k} \in \mathcal{N}_{c,d}^{k_1 \setminus k_2}} \lambda_{c'}^{k} \tag{7.49}$$

Note that if $\mathcal{N}_{c,d}^{k_1 \setminus k_2} = \emptyset$, then the right-hand side of the inequality is zero, and we have the case from the preprocessing in section 7.4.1. Hence, we only consider k_1 and k_2 where $\mathcal{N}_{c,d}^{k_1 \setminus k_2} \neq \emptyset$. Similar to the preprocessing we have to be careful when we add these constraints if we have a cyclic dominance as in the example of Figure 7.9. We get back to how we handle this case. We do not add the inequalities (7.49) to the pricing problem as the number of these constraints is $O\left(|\mathcal{K}_{c,d}|^2\right)$, instead we do an aggregation.

Consider again the course c and pattern $k_1 \in \mathcal{K}_{c,d}$. Let $\mathcal{V}' \subseteq \mathcal{K}_{c,d}$ be the set of patterns where $\mathcal{N}_{c,d}^{k_1 \setminus k'} \neq \emptyset$ for every $k' \in \mathcal{V}'$ and which are all dominated by k_1, i.e., for every $k' \in \mathcal{V}'$ we have that $\left(\alpha_{c,d}^{k'} - \overline{\pi}_{c,d}^{k'}\right) - \left(\alpha_{c,d}^{k_1} - \overline{\pi}_{c,d}^{k_1}\right) \geq \delta_{c,d}^{k_1,k'}$. In the neighbourhood $\mathcal{N}_{c,d}^{k_1}$ we let $\mathcal{N}_{c,d}^{k_1 \setminus \mathcal{V}'}$ denote the nodes that are connected to none of the nodes represented by the patterns in $\mathcal{V}'$. We illustrate this in Figure 7.11.

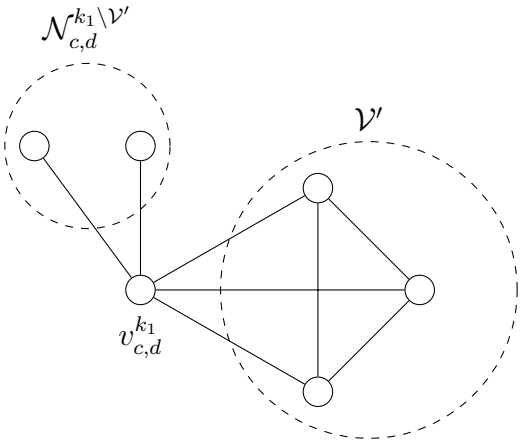


Figure 7.11: Illustration of the nodes in the pattern conflict graph $\mathcal{G}_d$ corresponding to the course c, the pattern k_1 and the set of patterns $\mathcal{V}'$. The set of nodes $\mathcal{N}_{c,d}^{k_1 \setminus \mathcal{V}'} \subseteq \mathcal{V}_d$ is the neighbourhood of the node $v_{c,d}^{k_1}$ excluding the neighbourhoods of the nodes in $\mathcal{V}'$ and all nodes corresponding to course c.

As for the pair of patterns in Figure 7.10, we consider any feasible solution where none of the nodes in $\mathcal{N}_{c,d}^{k_1 \setminus \mathcal{V}'}$ are selected, i.e., the corresponding variables are all set to zero. In this solution it is not beneficial to assign c to any of the patterns in $\mathcal{V}'$, as we can create a new solution by reassigning c to k_1 without increasing the objective value. So we can add the following inequality:

$$\sum_{v^{k'} \in \mathcal{V}'} \lambda_c^{k'} \leq \sum_{v_{c',d}^{k'} \in \mathcal{N}_{c,d}^{k \setminus \mathcal{V}'}} \lambda_{c'}^{k'} \tag{7.50}$$

As mentioned earlier we have to be careful about the dominance cycles when we add these inequalities. The way we get around this issue is to create a list L of all the patterns that we allow to be included in the left-hand-side of the inequality (7.50). Initially this list contains all the patterns of c, i.e., $L \leftarrow \mathcal{K}_{c,d}$. We then iterate through every pattern $k_1 \in \mathcal{K}_{c,d}$, remove k_1 from L and construct the set $\mathcal{V}'$. We then remove the patterns from $\mathcal{V}'$ that are not in the set L, i.e., $\mathcal{V}' \leftarrow \mathcal{V}' \cap L$. If $\mathcal{V}'$ is non-empty, then we add the inequality (7.50) and continue until all courses and patterns have been processed.

7.4.3 Local Branching

In this section we provide a brief introduction to Local Branching introduced by Fischetti and Lodi (2003) and how we apply it to the pricing problem. In our description of the local branching framework we focus on our implementation and do not cover every aspect of the techniques described by Fischetti and Lodi (2003), instead we refer to their article for details and a more general description.

Given a feasible solution $\overline{\lambda}$ let $\Delta\left(\lambda, \overline{\lambda}\right)$ be the distance between the solution $\overline{\lambda}$ and any other solution λ. We define the distance measurement to only count the number of binary variables

that change value from zero to one:

$$\Delta\left(\lambda, \overline{\lambda}\right) := \sum_{c \in \mathcal{C}, k \in \mathcal{K}: \overline{\lambda}_c^k = 0} \lambda_c^k$$

This distance measurement is referred to as the *symmetric hamming distance* (Fischetti and Lodi, 2003). Given a solution $\overline{\lambda}$ and an integer value k we can branch the problem into two subproblems:

$$\underbrace{\Delta\left(\lambda, \overline{\lambda}\right) \leq k}_{\text{(left branch)}} \quad \bigvee \quad \underbrace{\Delta\left(\lambda, \overline{\lambda}\right) \geq k+1}_{\text{(right branch)}}$$

In the basic local branching framework the left branch is solved by a generic MIP solver to optimality. If the optimal solution is an improvement of $\overline{\lambda}$, then this solution can be used to branch further on in the right branch.

When the basic local branching framework cannot find improving solutions, then Fischetti and Lodi (2003) describe methods to diversify. In its essence, the goal of the diversification is to find a new solution outside the neighbourhoods that have been explored. If such a solution can be found, then the local branching framework can be reapplied on that solution.

The question that remains is how to select the solutions on which to apply the local branching framework. The goal of the pricing problem is to find solutions with a negative objective value. So if we consider some solutions which almost have a negative objective value, then we can use them in the local branching framework.

Assume that we are in some iteration of the column generation algorithm where we have added columns to the master problem in previous iterations. As we are considering an optimal solution of the master problem, none of these columns has a negative reduced cost, but some of them might have a reduced cost of zero, e.g., the columns that are basic. The reduced cost of these columns corresponds to the objective value of the pricing problem to these solutions. Hence, the previously generated columns with a reduced cost of zero can be used as the solutions for the local branching framework.

In our implementation, we take all the previously generated columns with a reduced cost less than or equal to 10^{-5} (due to possible rounding errors). For each of these columns we check how many of the variables that are selected in the corresponding solution that has been fixed to zero in the preprocessing from section 7.4.1. If at least $k+1$ variables have been fixed to zero, then adding the local branching constraint makes the model infeasible. We then put the columns that do not lead to an infeasible model in a list ordered such that the first column in the list is the last one that was previously generated and the second column in the list is the second to last previously generated column. We use the first column in the list as the initial solution for the basic local branching framework, and the remaining columns are used for diversification. To illustrate this, consider an example of four previously generated feasible columns; $\overline{\lambda}^1$, $\overline{\lambda}^2$, $\overline{\lambda}^3$ and $\overline{\lambda}^4$. We denote these columns as 0-columns. In Figure 7.11a a two-dimensional representation of the solution space of the pricing problem is represented with the four 0-columns.

We first add $\Delta\left(\lambda, \overline{\lambda}^1\right) \leq k$ to the model and solve it to optimality. Assume that the optimal solution $\overline{\lambda}_1^1$ has a lower objective value than $\overline{\lambda}^1$. We then replace $\Delta\left(\lambda, \overline{\lambda}^1\right) \leq k$ by

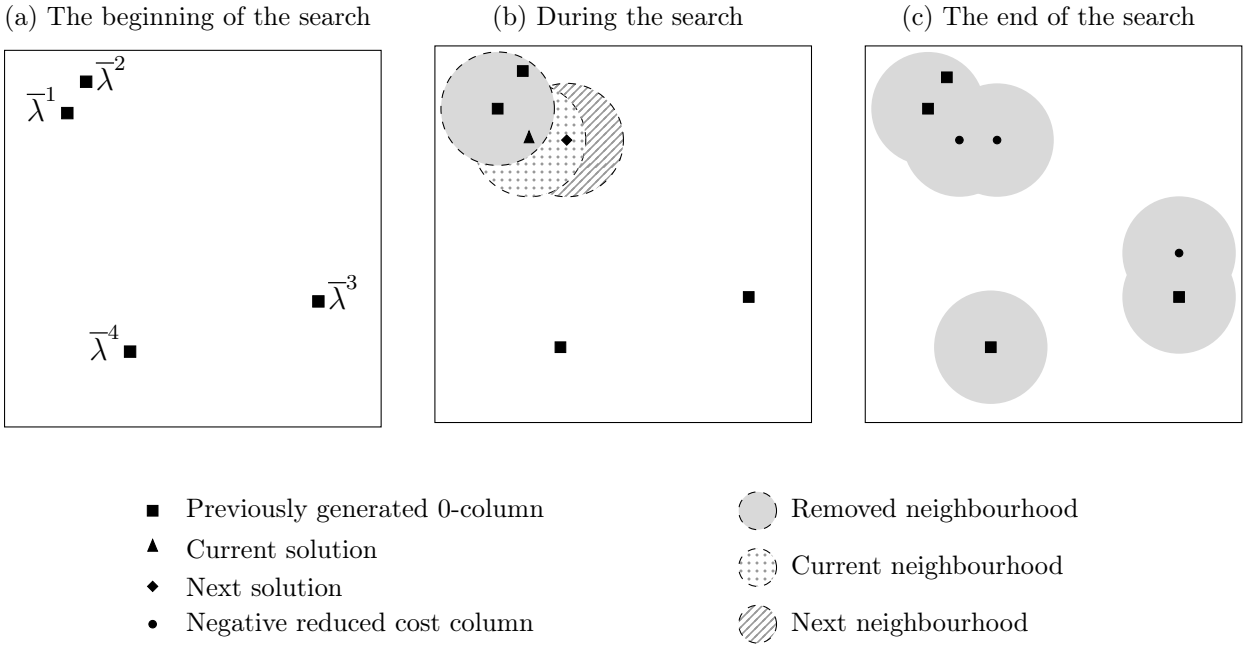


Figure 7.12: Illustration of the solution space of the example.

$\Delta\left(\lambda, \overline{\lambda}^1\right) \geq k+1$ and add $\Delta\left(\lambda, \overline{\lambda}^1_1\right) \leq k$ to the model. Again we solve the model to optimality and find the solution $\overline{\lambda}^1_2$. In Figure 7.11b the current state of the search is illustrated. The solution $\overline{\lambda}^1_1$ is illustrated as a triangle and denoted as the *current* solution. The solution $\overline{\lambda}^1_2$ is illustrated as a small diamond and denoted as the *next* solution.

After the neighbourhood of $\overline{\lambda}^1_2$ is explored, then no further improving solutions are found. The solution $\overline{\lambda}^2$ is skipped as it is inside the searched neighbourhoods. Next, the solution $\overline{\lambda}^3$, which is outside the neighbourhoods, is provided. The local branching framework is applied once again, leading to the improved solution $\overline{\lambda}^2_1$. No improving solutions are found in the neighbourhood of $\overline{\lambda}^2_1$, and so the neighbourhood of the last solution $\overline{\lambda}^4$ is searched. Here no improving solutions are found, and the local branching is stopped. In Figure 7.11c the solution space after running the local branching framework on all the 0-columns is illustrated. The squares mark the 0-columns, the dots mark the columns found with a negative reduced cost

If any columns with a negative reduced cost were found during the local branching search, then we add these to the master problem and stop. We do not search for solutions in the remaining solution space in this iteration of the column generation algorithm. If no columns with a negative reduced cost were generated, then we solve the model to optimality, with all the added right local branching constraints $\Delta\left(\lambda, \overline{\lambda}\right) \geq k+1$, and any columns found here with a negative reduced cost is added to the master problem.

7.5 Computational Results

In this section we describe our computational experiments and compare our results with other approaches from literature. We have conducted all tests on an Intel® Core™ i7-6700K CPU @

4.00GHz processor with four cores and eight logical processors and 32GB memory, running Windows 10. We use Gurobi version 7.0.1 provided by Gurobi Optimization, Inc. (2016) both for the master problem and in the pricing problems. We set the presolver to the most aggressive setting (Presolve=2), the number of threads to be equal to the number of processors (Threads=8) and the MIP gap to zero (MIPGap=0.0). The cut-off value is set to -10^{-5} (CUTOFF=-1e-5), such that the solver will not return solutions with objective values which are greater than -10^{-5}. We set the limit on the solutions to be returned to be the maximum possible (SolutionLimit=int.MaxValue) so we can extract all the columns that Gurobi generates with a negative reduced cost. The remaining parameters are set to their default value. We have implemented our code in C#, and we use the Parallel.ForEach method from the System.Threading.Tasks library to solve the pricing problems in parallel as they are independent.

We test our algorithm on 20 of the 21 data instances from ITC2007 named comp01 through comp21. The data instance that we do not include in the tests is comp11. The reason for not including this instance is the same reason that Cacchiani et al. (2013) do not include this instance in their tests, which is that the best-known upper bound is zero, and so we cannot improve the trivial lower bound of zero.

As the pricing problems that we solve are the same, except for the objective function, in each iteration, we build the model in Gurobi once and then change the objective function accordingly in each iteration. All the variables that are *removed* in the preprocessing phase described in section 7.4.1 are not removed from the model, but the upper bounds are set to zero instead. After the preprocessing the inequalities from section 7.4.2 are added, and then the local branching framework is provided with the model. All the calculations for the potentially isolated lectures in section 7.4.1 and 7.4.2 are performed when the pricing problems are built and the information is stored in a table so that we do not have to recalculate these values. We retrieve every column found by Gurobi that has a negative reduced cost and add them to the master problem. Afterwards, we remove all the constraints that we added in the pricing problems and set the upper bounds back to one.

We need to decide the value for k in the local branching framework. Fischetti and Lodi (2003) suggest to set it between 10 and 20, which in our case is between 5 and 10, as we are using the *symmetric hamming distance*. So we have tested the algorithm for $k = 5$ and $k = 10$. Furthermore, we have also tested for $k = 2$. The reason for testing for $k = 2$ is to mimic a 2-exchange heuristic (Wolsey, 1998). The total running time, including the time of building the model, the preprocessing and the enumeration of cliques, is reported in Table 7.2 for $k \in \{2, 5, 10\}$. The timings are reported in the format `hh:mm:ss` where `hh` is the amount of hours, `mm` is the amount of minutes and `ss` is the seconds. In the line (Total) the total amount of time spent is reported for each value of k and (Best) counts the number of times each value of k has the lowest running time.

In Table 7.2 we see that the algorithm is fastest for $k = 2$ regarding the total time to solve all instances. In the table we also see that in 13 out of the 20 instances the setting $k = 2$ is faster than for the other values. Thus, in the subsequent tests we report the results for $k = 2$.

In Table 7.3 we report the statistics of the column generation algorithm. For each data instance we report the number of iterations (Iter.). Then in the two following columns (Columns), we report the number of columns that were generated in total (Total) and how many of them that where generated by local branching (LocBra.). In the next four columns (Time) we report the timings of the algorithm. The total time spent by the algorithm is reported in the column

Table 7.2: The total time spent in the column generation algorithm for different values of k.

Instance	$k = 2$	$k = 5$	$k = 10$
comp01	**4:49**	4:56	5:06
comp02	41:42	49:49	**39:16**
comp03	**19:13**	26:11	31:34
comp04	13:25	**11:02**	14:01
comp05	**1:24:43**	3:09:19	4:27:34
comp06	**50:56**	1:21:40	1:14:51
comp07	**34:41**	35:22	38:58
comp08	**4:50**	13:22	12:58
comp09	25:02	**18:26**	20:14
comp10	1:06:11	48:22	**38:57**
comp12	**28:12:21**	30:33:25	41:45:45
comp13	**14:58**	28:25	20:38
comp14	15:16	**13:43**	18:37
comp15	**18:45**	22:32	31:27
comp16	57:43	**49:58**	55:07
comp17	**58:38**	1:37:46	1:50:46
comp18	**59:39**	1:08:27	1:47:18
comp19	**10:40**	20:53	18:37
comp20	1:49:58	1:58:21	**1:38:43**
comp21	**1:03:57**	1:34:33	1:42:48
Total	40:47:27	47:06:31	60:13:15
Best	**13**	**4**	**3**

(Total). The time spent on building the models, enumerating the cliques and preprocessing the pattern formulation is reported in the column (Build). The next two columns report the total time spent on solving the master problem (Master) and the total time spent on solving the pricing problems (Pricing). In the time spent on the pricing problem, the preprocessing, constraint generation and local branching is included. The timings are given in the format `hh:mm:ss` as in Table 7.2. The last column (Patterns Removed) reports the percentage of pattern variables that were removed in each iteration on average by the preprocessing in the pricing problems. The last line reports the averages of the columns generated by local branching compared to the total number of columns, the average time spent in each part compared to the total time and the average number of pattern variables removed compared to the total number of pattern variables in the pricing problems.

In Table 7.3 we see that the local branching framework is responsible for more than half of the columns generated on average. We also see that more than 90% of the total running time is spent on average on solving the pricing problems. Thus, more research on solution methods for the pricing problems is needed before the column generation algorithm can effectively be extended to a Branch & Price algorithm. Furthermore, we see that approximately 1% of the time is spent on average on solving the master problem, and lastly, 7.5% of the time is on average spent on building all the models and making the precalculations. We also see that

Table 7.3: Statistics of the column generation algorithm.

		Columns		Time				Patterns
Instance	Iter.	Total	LocBra.	Total	Build	Master	Pricing	Removed
comp01	37	1058	433	4:49	2:32	0	2:13	45.8%
comp02	191	13776	9160	41:42	1:55	18	39:13	40.1%
comp03	150	6762	1946	19:13	1:19	5	17:37	40.3%
comp04	227	7786	3821	13:25	56	8	12:03	50.7%
comp05	133	3909	2425	1:24:43	2:39	2	1:21:43	31.5%
comp06	284	20509	12076	50:56	1:34	1:01	47:52	48.1%
comp07	197	14451	6109	34:41	2:10	42	31:24	47.1%
comp08	239	6038	870	4:50	49	8	3:34	51.5%
comp09	208	10274	6914	25:02	56	9	23:42	46.3%
comp10	252	25381	18317	1:06:11	1:36	1:22	1:02:44	46.5%
comp12	301	12474	9794	28:12:21	3:27	31	28:07:14	28.6%
comp13	222	8641	3273	14:58	52	9	13:39	47.3%
comp14	180	10067	6079	15:16	1:10	10	13:40	47.3%
comp15	153	7119	2437	18:45	1:18	6	17:10	40.0%
comp16	236	20077	13168	57:43	1:34	48	54:55	49.8%
comp17	266	16640	8751	58:38	1:29	36	56:05	48.5%
comp18	231	5606	4205	59:39	1:22	5	57:42	46.1%
comp19	183	7857	4054	10:40	1:06	7	9:14	44.7%
comp20	307	32159	23802	1:49:58	1:55	2:34	1:44:50	46.3%
comp21	285	15251	5710	1:03:57	2:20	36	1:00:33	43.2%
Avg.			53.5%		7.5%	1.1%	90.1%	44.5%

almost half of the pattern variables are removed on average in each iteration of the algorithm. Next, compare the lower bounds that we obtained for the four open instances with the best-known bounds found reported on the website The Scheduling and Timetabling Research Group at the University of Udine, Italy (2015). We report the results in Table 7.4, where we have updated the best-known lower bounds with the bounds by Bagger et al. (2016). The last line in the table reports the average gap from the best-known upper bound.

Table 7.4: Comparison with the best-known bounds for the four open instances. *Updated value from Bagger et al. (2016).

	Best			DW	
Instance	UB	LB	Gap	LB	Gap
comp03	64	54*	16%	58	9%
comp05	284	211	26%	247	13%
comp12	294	175*	40%	248	16%
comp15	62	54*	13%	58	6%
Avg.			24%		11%

In Table 7.4 we see that our approach obtains a lower bound, which is an improvement of the best-known lower bound for all four of the open data instances. These improvements reduce

the average gap from the best-known upper bounds on these four instances from 24% to 11%.

In Table 7.5 we compare the lower bounds we obtained with the existing literature on the data instances comp01–comp10 and comp12–comp14, as these were the only data instances that were available for all the methods in literature. We compare the lower bounds obtained by our approach (DW) with BMPR10 (Burke et al., 2010b), BMPR12 (Burke et al., 2012), LL12 (Lach and Lübbecke, 2012), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013), BKSS16 (Bagger et al., 2017) and BDG16 (Bagger et al., 2016). If an article reports multiple lower bounds then we take the highest lower bound they obtain for each data instance. For each approach and each instance we mark the lower bound obtained in bold font if the bound is at least as good as the other approaches in the same table. If an approach obtains a lower bound which is better than all the other approaches in the same table then we mark it with an underline. In the table, we also report the average gap from the best-known upper bounds (Avg.). In the second last line (Best) we report the number of times when each approach obtains a lower bound which is at least as good as the other approaches. In the last line we report the number of times when each approach obtain a lower bound which is better than the other approaches. We use the same notation throughout all the tables.

Table 7.5: Comparison of the lower bounds for the different approaches.

Instance	UB	BMPR10	BMPR12	LL12	HB11	CCRT13	AN14	BKSS16	BDG16	DW
comp01	5	**5**	**5**	4	4	**5**	0	**5**	0	0
comp02	24	1	6	11	12	16	16	8	**24**	20
comp03	64	33	43	25	38	52	28	38	54	**58**
comp04	35	**35**	2	28	**35**	**35**	**35**	**35**	**35**	**35**
comp05	284	119	183	108	183	166	48	186	210	**247**
comp06	27	16	6	10	22	11	**27**	16	26	23
comp07	6	**6**	0	**6**	**6**	**6**	**6**	**6**	**6**	**6**
comp08	37	**37**	2	**37**	**37**	**37**	**37**	**37**	**37**	**37**
comp09	96	68	0	46	72	92	35	74	**96**	92
comp10	4	**4**	0	**4**	**4**	2	**4**	**4**	**4**	**4**
comp12	294	101	7	53	109	100	99	142	175	**248**
comp13	59	54	0	41	**59**	57	**59**	**59**	**59**	**59**
comp14	51	42	0	46	**51**	48	**51**	44	**51**	49
Avg.		28.0%	77.5%	35.0%	19.4%	21.7%	31.0%	20.8%	14.3%	13.7%
Best		**5**	**1**	**3**	**6**	**4**	**7** **1**	**6**	**8** **2**	**8** **3**

In Table 7.5 we see that our approach obtains a lower bound which is at least as good as the lower bounds of the other approaches on eight of the instances. On three of these instances the lower bound we obtain is better than for the other approaches. Furthermore, we see that our approach has the lowest average gap (13.7%) to the best-known upper bounds. In Table 7.6 we

compare our results on all 20 data instances to HB11, CCRT13, AN14, BKSS16 and BDG16 since they report results for all these instances.

Table 7.6: Comparison of the lower bounds for the different approaches for all 20 data instances.

Instance	UB	HB11	CCRT13	AN14	BKSS16	BDG16	DW
comp01	5	4	**5**	0	**5**	0	0
comp02	24	12	16	16	8	**24**	20
comp03	64	38	52	28	38	54	**58**
comp04	35	**35**	**35**	**35**	**35**	**35**	**35**
comp05	284	183	166	48	186	210	**247**
comp06	27	22	11	**27**	16	26	23
comp07	6	**6**	**6**	**6**	**6**	**6**	**6**
comp08	37	**37**	**37**	**37**	**37**	**37**	**37**
comp09	96	72	92	35	74	**96**	92
comp10	4	**4**	2	**4**	**4**	**4**	**4**
comp12	294	109	100	99	142	175	**248**
comp13	59	**59**	57	**59**	**59**	**59**	**59**
comp14	51	**51**	48	**51**	44	**51**	49
comp15	62	38	52	28	38	54	**58**
comp16	18	16	13	**18**	13	**18**	17
comp17	56	48	48	**56**	44	53	**56**
comp18	61	24	**52**	27	36	**52**	**52**
comp19	57	56	48	46	56	**57**	51
comp20	4	2	**4**	**4**	0	**4**	3
comp21	74	61	68	42	57	**74**	71
Avg.		22.3%	19.0%	28.8%	26.2%	10.9%	12.2%
Best		**6**	**6**	**10** **1**	**6**	**13** **4**	**11** **4**

In Table 7.6 we see that our approach obtains a lower bound which is at least as good as the lower bound obtained by the other approaches on 11 of the instances. On four of these instances, our approach obtains a better lower bound than the other approaches. These four instances are the open instances. Lastly, we see that our approach obtains the second lowest average gap (12.2%) to the best-known upper bounds, where the original pattern formulation has the lowest average gap (10.9%).

As we solved the LP-relaxation of the Dantzig-Wolfe master problem, it could be interesting to do more research in extending the algorithm into a full Branch & Price algorithm. It could also be interesting to derive cutting plane techniques that can be incorporated in the column generation algorithm to push the bounds even further as our approach shows great potential. As the local branching is a generic method, it can be used in pricing problems for other column generation algorithms. To the best of our knowledge, this work is the first implementation of local branching in the pricing problem for a column generation algorithm. We think that many

pricing problems, in general, fits well with local branching since every iteration of the column generation algorithm provides new solutions to be used in the framework.

7.6 Conclusion

In this article we applied a Dantzig-Wolfe Decomposition on a pattern formulation for the Curriculum-based Course Timetabling problem. The pattern formulation is based on enumerating all the time schedules to which the courses can be assigned each day. The pattern formulation only considers the time schedule of the problem. Thus, the formulation is a lower bounding method for the original problem. The decomposition resulted in a pricing problem for each day, where each pricing problem generates a schedule for an entire day. We showed that the pricing problem contained a special structure which we exploited in a preprocessing phase. We then showed how the preprocessing technique could be used to derive inequalities for the pricing problem. Lastly, we described how we applied Local Branching to solve the pricing problem. To the best of our knowledge, this is the first time Local Branching is implemented in a pricing problem, but it is general enough to be applied in other column generation algorithms. We tested our algorithm on 20 data instances used in the second International Timetabling Competition. On these instances the preprocessing technique we applied removed more than 40% of the pattern variables from the pricing problem on average. We compared the lower bounds that we obtained with other approaches from literature. In 11 of the instances our algorithm obtained a lower bound which was at least as good as the other approaches. Four of these instances are still *open*, meaning that the best-known upper bound does not equal the best-known lower bound. In all of these four instances our algorithm improved the lower bound, such that the average gap was decreased from 24% to 11%. We showed that more than 90% of the total time of the algorithm were spent on solving the pricing problem, and we concluded that more research is needed in the pricing problems before the algorithm is extended into a full Branch & Price Algorithm.

Acknowledgments

The authors would like to thank Guy Desaulniers and Jacques Desrosiers for their contribution to the previous work on which this paper is based. The authors would also like to thank Christina Scheel Persson for her valuable feedback to improve the manuscript.

Funding: This work is part of an industrial PhD project funded by Innovation Fund Denmark (IFD). IFD has supported this work solely financially and has not taken part in any research related activities.

References

Asín Aschá, R. and Nieuwenhuis, R. (2014). "Curriculum-based course timetabling with SAT and MaxSAT". In: *Annals of Operations Research* 218, pp. 71–91.

Bagger, N.-C. F., Kristiansen, S., Sørensen, M., and Stidsen, T. R. (2017). "Flow Formulations for Curriculum-based Course Timetabling". In: *Submitted to Annals of Operations Research. Preprint available at Optimization-Online.org*. URL: http://www.optimization-online.org/DB_HTML/2016/12/5786.html.

Bagger, N.-C. F., Desaulniers, G., and Desrosiers, J. (2016). "Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem". In: *Submitted to Journal of Scheduling. Preprint available from Les Cahiers du GERAD*. URL: https://www.gerad.ca/en/papers/G-2016-71.

Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. (2015). "An overview of curriculum-based course timetabling". In: *TOP* 23.2, pp. 313–349.

Bonutti, A., De Cesco, F., Di Gaspero, L., and Schaerf, A. (2012). "Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization, and results". In: *Annals of Operations Research* 194.1, pp. 59–70.

Bron, C. and Kerbosch, J. (1973). "Algorithm 457: Finding All Cliques of an Undirected Graph". In: *Communications of the ACM* 16.9, pp. 575–577. DOI: 10.1145/362342.362367. URL: http://doi.acm.org/10.1145/362342.362367.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2008). "Penalising Patterns in Timetables: Novel Integer Programming Formulations". In: *Operations Research Proceedings 2007*. Springer, pp. 409–414.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010a). "A supernodal formulation of vertex colouring with applications in course timetabling". In: *Annals of Operations Research* 179.1, pp. 105–130.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2010b). "Decomposition, reformulation, and diving in university course timetabling". In: *Computers & Operations Research* 37.3, pp. 582–597. DOI: 10.1016/j.cor.2009.02.023.

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). "A branch-and-cut procedure for the Udine Course Timetabling problem". In: *Annals of Operations Research* 194.1, pp. 71–87.

Cacchiani, V., Caprara, A., Roberti, R., and Toth, P. (2013). "A new lower bound for curriculum-based course timetabling". In: *Computers and Operation Research* 40.10, pp. 2466–2477. DOI: 10.1016/j.cor.2013.02.010.

Desrosiers, J. and Lübbecke, M. E. (2010). "A Primer in Column Generation". In: *Column Generation*. Ed. by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Springer Science+Business Media, Inc. Chap. 1, pp. 1–32. ISBN: 978-1-4419-3799-5.

Di Gaspero, L., Schaerf, A., and McCollum, B. (2007). "The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3) — preliminary presentation —". In: *Association for the Advancement of Artificial Intelligence (www.aaai.org)*.

Fischetti, M. and Lodi, A. (2003). "Local branching". In: *Mathematical Programming* 98.1-3, pp. 23–47. DOI: 10.1007/s10107-003-0395-5.

Gurobi Optimization, Inc. (2016). *Gurobi Optimizer Reference Manual*. URL: http://www.gurobi.com.

Hao, J. K. and Benlic, U. (2011). "Lower bounds for the ITC-2007 curriculum-based course timetabling problem". In: *European Journal of Operational Research* 212.3, pp. 464–472.

Lach, G. and Lübbecke, M. E. (2008). "Optimal University Course Timetables and the Partial Transversal Polytope". In: *International Workshop on Experimental and Efficient Algorithms*. Springer, pp. 235–248.

Lach, G. and Lübbecke, M. E. (2012). "Curriculum based course timetabling: New solutions to Udine benchmark instances". In: *Annals of Operations Research* 194.1, pp. 255–272.

Lübbecke, M. E. (2015). "Comments on: An Overview of Curriculum-Based Course Timetabling". In: *TOP* 23.2, pp. 359–361.

Martin, R. K. (1999). *Large scale linear and integer optimization: a unified approach*. Kluwer Academic Publishers.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Di Gaspero, L., Qu, R., and Burke, E. K. (2010). "Setting the research agenda in automated timetabling: The second international timetabling competition". In: *INFORMS Journal on Computing* 22.1, pp. 120–130.

The Scheduling and Timetabling Research Group at the University of Udine, Italy (2015). *Curriculum-Based Course TimeTabling*. Last retrieved October 2015, http://tabu.diegm.uniud.it/ctt/index.php. URL: http://tabu.diegm.uniud.it/ctt/index.php.

Wolsey, L. A. (1998). "Heuristic Algorithms". In: *Integer Programming*. Ed. by R. L. Graham, J. K. Lenstra, and R. E. Tarjan. John Wiley & Sons, Inc. Chap. 12, pp. 203–220. ISBN: 0-471-28366-5.